CORE
Metadata, citation and similar papers at core.ac.uk
Provided by Repositório Científico do Instituto Politécnico do Porto

Instituto Superior de Engenharia do Porto

Construção de interfaces em Flex para sistemas de experimentação remota

Pedro José Lima Teixeira

Dissertação para obtenção do Grau de Mestre em

Engenharia Informática

Área de Especialização em

Arquitecturas, Sistemas e Redes

Orientador: Doutor Manuel Gradim de Oliveira Gericota

Co-orientador: Mestre Ricardo Jorge Guedes da Silva Nunes da Costa

Júri:

Presidente:

Prof.ª Doutora Maria de Fátima Coutinho Rodrigues, Professora Coordenadora do Departamento de Engenharia Informática do Instituto Superior de Engenharia do Porto

Vogais:

Prof. Doutor José Manuel de Magalhães Cruz, Professor Auxiliar do Departamento de Engenharia Informática da Faculdade de Engenharia da Universidade do Porto

Prof. Doutor Manuel Gradim de Oliveira Gericota, Professor Adjunto do Departamento de Engenharia Electrotécnica do Instituto Superior de Engenharia do Porto

Mestre Ricardo Jorge Guedes da Silva Nunes da Costa, Equiparado a Assistente do Departamento de Engenharia Electrotécnica do Instituto Superior de Engenharia do Porto

Porto, Setembro 2010

Agradecimentos

A escrita de uma dissertação implica um grau de exigência e esforço elevado para o autor. No entanto, também deve ser dado valor àqueles que, de uma forma ou de outra, contribuíram para que os resultados finais fossem alcançados.

Começo por agradecer ao LABORIS por me ter dado a oportunidade de desenvolver o trabalho que levou à escrita desta dissertação. Aos meus orientadores, Manuel Gericota e Ricardo Costa, o meu agradecimento pela disponibilidade demonstrada, pelas sugestões e pelo espírito crítico que tanto é necessário no desenvolvimento de um sistema deste tipo.

Aos meus colegas de Mestrado, principalmente ao Valentim Sousa, Rui Eusébio, Armindo Felgueiras, Rui Felgueiras, Flávio Oliveira, Joaquim Teixeira e Sílvio Lopes, que durante estes dois anos me acompanharam neste duro percurso e me incentivaram sempre a dar o melhor de mim, o meu muito obrigado.

Às empresas pelas quais passei enquanto frequentei o Mestrado, *Vodafone Portugal - Comunicações Pessoais, S.A.*, *8W Media - Design, Comunicação e Tecnologia* e *Alert Life Sciences Computing S.A.*, e aos meus colegas de trabalho, um agradecimento por toda a compreensão e incentivos demonstrados.

O meu muito obrigado à minha família, principalmente aos meus pais e à minha irmã. A exigência de trabalhar e estudar ao mesmo tempo levou a que não lhes prestasse a atenção necessária e que eles mereciam. O seu apoio e compreensão foram fundamentais ao longo destes dois anos.

O meu agradecimento a quem, não estando directamente referenciado, também contribuiu para a realização desde trabalho.

Resumo

Os laboratórios de experimentação remota estão normalmente associados a tecnologias ou soluções proprietárias, as quais restringem a sua utilização a determinadas plataformas e obrigam ao uso de *software* específico no lado do cliente.

O ISEP possui um laboratório de experimentação remota, baseado em instrumentação virtual, usado no apoio ao ensino da electrónica e construído sobre uma plataforma NI-ELVIS da National Instruments. O *software* de controlo da plataforma recorre à linguagem gráfica de programação LabVIEW. Esta é uma ferramenta desenvolvida pela National Instruments que facilita o desenvolvimento de aplicações de sistemas de experimentação remota, mas que possui várias limitações, nomeadamente a necessidade de instalação do lado do cliente de um *plug-in*, cuja disponibilidade se encontra limitada a determinadas versões de sistemas operativos e de *Web Browsers*.

A experiência anterior demonstrou que estas questões limitam o número de clientes com possibilidade de acesso ao laboratório remoto, para além de, em alguns casos, se ter verificado não ser transparente a sua instalação e utilização.

Neste contexto, o trabalho de investigação consistiu no desenvolvimento de uma solução que permite a geração de interfaces que possibilitam o controlo remoto do sistema implementado, e que, ao mesmo tempo, são independentes da plataforma usada pelo cliente.

Palavras-chave: LabVIEW, Adobe Flex, VI, PHP, Web Service, NI-ELVIS

Abstract

The remote experimentation laboratories are usually associated with technology or proprietary solutions, which restrict its use to certain platforms and require the use of specific software on the client side.

ISEP has a remote experimentation laboratory, based on virtual instrumentation, built on a NI-ELVIS platform, from National Instruments, and used to support the teaching of electronics. The control software of the platform uses the graphical programming language LabVIEW. This is a tool developed by National Instruments that facilitates the development of applications aimed at remote experimentation systems. However, it has several limitations, including the need for client-side installation of a plug-in, whose availability is limited to certain versions of operating systems and Web Browsers.

Past experience has shown that these issues limit the number of customers with possibility to access the remote laboratory. In some cases, its installation and use has shown not to be transparent to the user.

In this context, the aim of this research work is the generation of interfaces independent of the platform used by the client, allowing the remote control of the implemented remote laboratory.

Keywords: LabVIEW, Adobe Flex, VI, PHP, Web Service, NI-ELVIS

Índice

Agradecimentos i
Resumo iii
Abstract v
Índice vii
Índice de Figuras ix
Índice de Tabelas xi
Acrónimos xiii
1. Introdução 1
1.1. Contextualização e objectivos 1
1.2. Calendarização 2
1.3. Estrutura do documento 3
2. Conceitos e Estado da Arte 5
2.1. Sistemas de experimentação remota 5
2.2. Plataformas de experimentação remota 12
2.3. Tecnologias de construção de interfaces gráficas 15
2.3.1. JavaFX 16
2.3.2. Microsoft Silverlight 18
2.3.3. Adobe Flex 19
3. Da identificação de requisitos às opções de implementação 23
3.1. A não universalidade da solução actual 23
3.2. Identificação de requisitos da nova solução 26
3.3. Soluções técnicas e opções 28
3.3.1. Flex como tecnologia para a construção da interface 29

3.3.2. Versão do LabVIEW 30
3.3.3. Ligação entre *Middleware* e LabVIEW 31
3.3.4. Middleware 33
3.3.5. Ligação entre *Middleware* e Interface 35
4. Implementação da nova solução 39
4.1. Arquitectura da solução 39
4.2. Implementação do laboratório 40
4.2.1. Plataforma LabVIEW 2009 40
4.2.2. Lógica de negócio 48
4.2.3. Interface Flex 55
4.3. Controlo de acessos e segurança 63
4.3.1. Controlo de acessos 63
4.3.2. Acesso aos serviços PHP 65
4.3.3. Código fonte 66
4.3.4. Outras configurações 66
5. Demonstração de resultados 69
6. Conclusão 75
6.1. Objectivos alcançados 75
6.2. Trabalho futuro 76
6.3. Considerações finais 77
Referências Bibliográficas 79
ANEXOS 85
ANEXO I – Guião do trabalho proposto aos alunos 87
ANEXO II – Anexos digitais 93

Índice de Figuras

Figura 1 - Estrutura geral dos laboratórios remotos [Hanson et al., 2009] 6

Figura 2 - Arquitectura do sistema ReLOAD [Hanson et al., 2009] 8

Figura 3 - Arquitectura da plataforma eLab [Lewis et al., 2009] 9

Figura 4 - Arquitectura do sistema JIL 11

Figura 5 - *Front Panel* e *Block Diagram* de uma aplicação em LabVIEW 13

Figura 6 - Estação NI-ELVIS 14

Figura 7 - Quota de mercado das plataformas Flash, Java e Silverlight 16

Figura 8 - Versão do *plug-in* instalado por cada plataforma 16

Figura 9 - Arquitectura geral do JavaFX 17

Figura 10 - Arquitectura geral do Silverlight [MSDN, 2010] 19

Figura 11 - Arquitectura geral da Flex Framework [Brown, 2007] 20

Figura 12 - Visão geral da plataforma Flex [Gassner, 2010] 21

Figura 13 - *Workflow* da solução existente 24

Figura 14 - Arquitectura implementada para o laboratório remoto em funcionamento 25

Figura 15 - Diagrama de casos de utilização 28

Figura 16 - Arquitectura simplificada do novo laboratório remoto 29

Figura 17 - Arquitectura simplificada do novo laboratório remoto: Interface 29

Figura 18 - Arquitectura simplificada do novo laboratório remoto: LabVIEW 30

Figura 19 - Arquitectura simplificada do novo laboratório remoto: *Middleware* ↔ LabVIEW 31

Figura 20 - Arquitectura simplificada do novo laboratório remoto: *Middleware* 33

Figura 21 - Arquitectura simplificada do novo laboratório remoto: *Middleware* ↔ Interface 35

Figura 22 - Tempo gasto por cada meio de comunicação 37

Figura 23 - Volume de dados gasto por cada meio de comunicação 37

Figura 24 - Arquitectura geral do novo laboratório remoto 39
Figura 25 - *Block Diagram* alterado para os valores obtidos da estação NI-ELVIS 41
Figura 26 - *Block Diagram* alterado para os valores de configuração 42
Figura 27 - *Front Panel* (esquerda) e *Block Diagram* (direita) do VI sharedchana.vi 45
Figura 28 - *Connector Pane* na construção de um *Web Service* 45
Figura 29 - *Front Panel* com o *Connector Pane* visível e as variáveis definidas 46
Figura 30 - *Block Diagram* (em cima) e *Connector Pane* (em baixo) do VI “data.vi” 46
Figura 31 - Estrutura de ficheiros em PHP 51
Figura 32 - *Workflow* do método "submitRemoteLab" 53
Figura 33 - *Workflow* do ficheiro "callvi.php" 54
Figura 34 - *Workflow* do método "resultRemoteLab" 55
Figura 35 - Ciclo de vida de uma acção em Mate 62
Figura 36 - Arquitectura simplificada do sistema de controlo de acessos 64
Figura 37 - Estação NI-ELVIS do laboratório remoto 69
Figura 38 - Esquema do circuito montado no laboratório remoto 70
Figura 39 - Interface geral do novo Laboratório Remoto 71
Figura 40 - Balão de ajuda a avisar da gama de valores 72
Figura 41 - Aviso a informar de valores inválidos 72
Figura 42 - Aviso de ligação em curso 73
Figura 43 - Interface com os dados apresentados 73

Índice de Tabelas

Tabela 1 - Plano de trabalhos do trabalho efectuado 2

Tabela 2 - Cruzamento entre as formas de comunicação e suas características.....33

Tabela 3 - Cruzamento das plataformas com as características para o *Middleware*.....35

Tabela 3 - Lista de variáveis partilhadas criadas 43

Tabela 4 - VIs criados para a construção do *Web Service* 44

Tabela 5 - Lista dos URL *Mappings* definidos para o *Web Service* 48

Tabela 7 - Estrutura da tabela “requisition” da base de dados “remotelab”.....50

Tabela 8 - Cruzamento entre as *frameworks* Flex de MVC e as suas características.....59

Tabela 9 - Estrutura do projecto RemoteLab em Flex a partir da pasta do código fonte.....60

Acrónimos

AMF	Action Message Format
AMFPHP	Action Message Format for PHP
CSS	Cascading Style Sheets
DEI	Departamento de Engenharia Informática
ELVIS	Educational Laboratory Virtual Instrumentation Suite
FPGA	Field Programmable Gate Array
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
IDE	Integrated Development Environment
ISEP	Instituto Superior de Engenharia do Porto
IPP	Instituto Politécnico do Porto
JRE	Java Runtime Environment
JSON	JavaScript Object Notation
LABORIS	Laboratório de Investigação em Sistemas de Teste
LabVIEW	Laboratory Virtual Instrument Engineering Workbench
LAMP	Linux, Apache, MySQL, PHP
MVC	Model View Controller
MXML	Macromedia FleX Markup Language
NI	National Instruments
PC	Personal Computer
PHP	PHP: Hypertext Preprocessor
REST	Representational State Transfer
RIA	Rich Internet Application

SDK	Software Development Kit
SOAP	Simple Object Access Protocol
SSO	Single Sign On
TCP	Transmission Datagram Protocol
UDP	User Datagram Protocol
VI	Virtual Instrument
VM	Virtual Machine
XAML	eXtensible Application Markup Language
XAMP	X, Apache, MySQL, PHP
XML	Extensible Markup Language
WAMP	Windows, Apache, MySQL, PHP
WWW	World Wide Web

1. Introdução

Esta dissertação apresenta o trabalho desenvolvido no âmbito da Tese de Mestrado em Engenharia Informática. O desenvolvimento do tema proposto está descrito ao longo do documento, onde são igualmente apresentados todos os passos seguidos e justificadas todas as opções tomadas.

1.1. Contextualização e objectivos

Os sistemas de experimentação remota têm ganho especial ênfase nos últimos anos, aproveitando a explosão da Internet para estarem cada vez mais presentes nas instituições de ensino. Há um par de anos, foi implementado um sistema de experimentação remota no ISEP, que permite aos alunos aceder a um laboratório de experimentação remoto, ou laboratório remoto, destinado ao ensino da electrónica, através de um *Web Browser*. No entanto, esse sistema possui algumas desvantagens, sendo a principal o facto de os alunos necessitarem de instalar um *plug-in* fornecido pela National Instruments para conseguirem visualizar no seu computador a interface que lhes permite interagir com o laboratório remoto. A disponibilidade dessa aplicação encontra-se limitada a certas versões de sistemas operativos e de *Web Browsers* e o seu funcionamento nem sempre é linear. Por outro lado, obriga à abertura de um determinado número de portas nas *firewalls*.

Nesse contexto, surgiu a ideia de criar um laboratório remoto cuja utilização fosse mais transparente para o utilizador, e que resolvesse os problemas existentes no antigo laboratório. Os objectivos a alcançar nesta nova implementação exigem que a interface seja construída a partir das recentes plataformas existentes no mercado, que possibilitam uma construção rica em elementos de visualização, e que seja mais transparente e linear do ponto de vista do utilizador, mantendo, no entanto, todas as funcionalidades presentes na solução existente.

Para a implementação desse novo laboratório remoto, foi necessária uma vasta investigação sobre os sistemas remotos que existem actualmente disponíveis, assim como de algumas soluções em funcionamento já implementadas em instituições de ensino. Com base nessa investigação, foi projectada a arquitectura do novo laboratório remoto, que possui várias camadas e usa diversas plataformas, na sua maioria *open source*, como o PHP ou o Adobe Flex. Com base numa investigação para avaliar a viabilidade do uso dessas plataformas na aplicação, partiu-se para o desenvolvimento da aplicação e

respectiva implementação. Por fim, foram realizados vários ensaios ao novo laboratório remoto e analisados e comparados os resultados de funcionamento com os obtidos com a anterior solução.

O novo laboratório remoto permitiu evitar o uso do *plug-in* proprietário da National Instruments na interface que interage com o laboratório. Em detrimento deste é usado o *plug-in* do Flash Player, que está disponível na quase totalidade dos sistemas operativos e *Web Browsers*. Para além disso, houve uma diminuição drástica do tempo de carregamento da interface, sendo que os problemas com as *firewalls* foram igualmente resolvidos.

1.2. Calendarização

Foi elaborado um plano de trabalhos de modo a manter organizado o trabalho e executar pela ordem certa as tarefas. Na Tabela 1 estão descritas todas as tarefas efectuadas ao longo dos vários períodos temporais.

Tabela 1 - Plano de trabalhos do trabalho efectuado

Período temporal	Descrição das tarefas
Outubro de 2009 Novembro de 2009 Dezembro de 2009	- Ambientação com o projecto - Levantamento do Estado da Arte
Janeiro de 2010 Fevereiro de 2010	- Investigação sobre o funcionamento dos sistemas de experimentação remota - Investigação da linguagem de programação LabVIEW e sua conectividade com outras plataformas
Março de 2010	- Elaboração de protótipos - Definição dos requisitos e arquitectura a adoptar
Abril de 2010 Maio de 2010	- Implementação do projecto - Início da elaboração do documento final (dissertação)
Junho de 2010 Julho de 2010	- Fase de testes e correcção de *bugs* da aplicação - Elaboração do documento final (dissertação)
Agosto de 2010 Setembro de 2010	- Conclusão e correcção do documento final (dissertação)

1.3. Estrutura do documento

Esta dissertação está dividida em **cinco** capítulos principais.

Neste **primeiro capítulo** efectua-se o enquadramento desta dissertação, sendo apontados os principais objectivos e delineado o seu planeamento.

No **segundo capítulo** está descrita toda a investigação efectuada sobre laboratórios remotos já existentes, assim como a definição de alguns termos relacionados com ambientes remotos e construção de interfaces.

A não universalidade da solução actual está descrita no **terceiro capítulo**. Para além disso, são também identificados todos os requisitos a implementar no novo laboratório remoto e justificadas todas as opções efectuadas para a sua implementação.

No **quarto capítulo** descreve-se a arquitectura do novo laboratório remoto, assim como todo o processo de desenvolvimento.

No **quinto capítulo** é exemplificado o funcionamento do novo laboratório remoto com recurso a um trabalho sobre o uso de amplificadores operacionais na construção de uma fonte de tensão regulada.

Por fim, as conclusões do trabalho efectuado estão apresentadas no **sexto capítulo**, onde também são sugeridas algumas propostas para um trabalho futuro.

2. Conceitos e Estado da Arte

2.1. Sistemas de experimentação remota

A necessidade de modernização dos métodos de ensino, aliada à contínua investigação nos centros universitários, tem provocado uma profunda mudança nas abordagens que as instituições de ensino seguem em relação aos métodos usados para ensinar. Os recursos humanos, físicos e financeiros tendem a ser minimizados no decorrer do desenvolvimento tecnológico da informática. Desta forma, o *e-learning* tem centrado as atenções dos investigadores, pois oferece várias vantagens, para alunos, professores e para a instituição. Seguindo essa linha, já não faz sentido haver barreiras físicas no que se refere à comunicação entre alunos e professores, ou mesmo entre estes e os laboratórios escolares, pois com o uso global da Internet passou a ser possível aceder a qualquer recurso virtual, e mesmo real, como é o caso dos laboratórios remotos, desde qualquer localização.

Os laboratórios remotos, ou sistemas de experimentação remota, tal como o próprio nome indica, são sistemas que se caracterizam por estarem disponíveis remotamente para os utilizadores através da Internet sem necessidade de presença física e a maioria das vezes sem restrições de horários de utilização.

Os sistemas de experimentação remota têm ganho especial ênfase durante os últimos anos, embalados pela expansão da Internet. Costumam ser bem aceites pelos utilizadores por serem sistemas inovadores, entusiasmantes, motivantes, inspiradores e fáceis de usar. Estes sistemas também são usados na química, na mecânica, na electrotécnica, entre outras. No caso específico da informática, e para uma maior disponibilidade, convém que o sistema seja acessível de qualquer local (preferencialmente através de um *Web Browser*) e o mais multiplataforma possível, para que utilizadores com sistemas operativos diferentes possam aceder nas mesmas condições [Sousa et al., 2010].

No entanto, nem sempre é fácil construir um sistema deste tipo, pois requer que existam condições físicas, monetárias e até administrativas para que seja implementado. Para além disso, trata-se de um sistema crítico, em que o *hardware* é sujeito a factores externos, como, por exemplo, as condições meteorológicas ou a perturbações da rede eléctrica.

A Figura 1 caracteriza a arquitectura típica de um sistema deste tipo. Basicamente, existe uma interface com a qual o utilizador interage, e que comunica com o laboratório remoto.

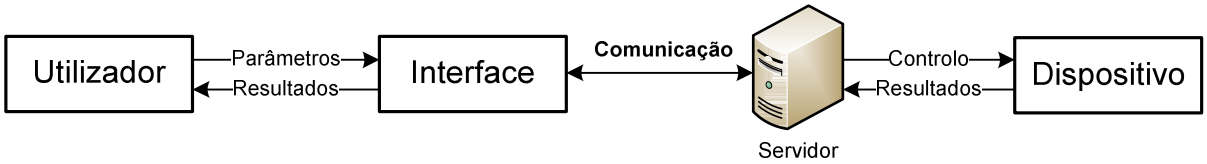

Figura 1 - Estrutura geral dos laboratórios remotos [Hanson et al., 2009]

O facto de existir uma interface virtual com a qual o utilizador interage, faz com que haja uma grande preocupação na eliminação das barreiras físicas previamente mencionadas. Isto é, há a necessidade de reproduzir virtualmente o laboratório físico através de fotografias, vídeos e outros elementos que dêem a percepção de um laboratório real, de forma a transmitir ao utilizador um sentimento de estar presente e mais familiarizado com o sistema. Nem todas as áreas necessitam de ter o laboratório virtual dessa forma, mas outras, como na medicina, os profissionais têm mais contacto com as pessoas e necessitam de ter mais presente um ambiente que simule o que existe na realidade [Hanson et al., 2009].

As principais funcionalidades ou requisitos que um laboratório remoto deve ter são [Lewis et al., 2009]:

- Disponibilidade: O sistema deve estar sempre disponível para responder aos pedidos dos utilizadores, mesmo quando existem situações de acesso em simultâneo. Normalmente sugere-se a implementação de uma lista de espera FIFO (*First In First Out* em inglês, que significa o primeiro pedido a chegar é o que é tratado em primeiro lugar) para gerir correctamente os pedidos;

- Segurança: Com foco em três pontos fortes: a validação dos dados introduzidos pelo utilizador; o protocolo de autenticação do utilizador; e o controlo de acesso indevido aos dados;

- Flexível: Capacidade de se adaptar a novos ambientes, como por exemplo, mudar a lógica da experiência, suportar diversas ligações em simultâneo com cenários diferentes ou capacidade de mudar de laboratório (caso existam vários);

- Portabilidade: Capacidade de aceder ao laboratório remoto de qualquer sistema operativo ou *Web Browser*;

- Custo de manutenção reduzido: Se possível, devem-se usar linguagens *open source* na implementação, tendo em vista uma manutenção futura ou evitar a compra de licenças.

Uma grande vantagem dos sistemas de experimentação remota é o facto de os utilizadores possuírem maior liberdade de interagir com o sistema. Isto faz com que os níveis de atenção sejam muito superiores em relação aos laboratórios tradicionais, que normalmente possuem alguém a explicar algum processo e têm uma plateia inteira a olhar para o orador. Outra grande vantagem dos ambientes remotos, é o facto de permitir aos utilizadores acederem ao laboratório, sem estarem fisicamente presentes, durante 24 horas por dia, 7 dias por semana, seja feriado ou fim-de-semana. Na educação é muito vantajoso para um aluno que está, por exemplo, doente. Para além disso, podem repetir a experiência as vezes que queiram, resolvendo possíveis erros de anteriores submissões [Costa, 2003].

Nota-se que este tipo de laboratórios eliminam barreiras e mitos do ensino tradicional, fomentando o trabalho colaborativo, pois vários alunos podem estar a aceder remotamente ao laboratório e discutir os resultados em *chats* ou por telefone.

A arquitectura nos sistemas de experimentação remota segue, normalmente, a mesma estrutura, isto é, um dispositivo comunica com um computador remoto, podendo ainda haver uma camada de *middleware* que implementa a lógica da aplicação. As aplicações de *software* que são necessárias servem para conectar o dispositivo com o computador local, o computador local com o remoto, e o computador remoto com o utilizador (interface com o utilizador). Para isso, são usadas várias linguagens de programação na elaboração deste tipo de sistemas, algumas delas proprietárias (LabVIEW e Matlab-Simulink) e outras tradicionais (Java ou PHP) para a implementação da lógica da aplicação [Lewis et al., 2009].

Existem vários exemplos de adopção de laboratórios de experimentação remota na área da educação. A seguir vão ser expostos vários exemplos de laboratórios de experimentação remota já em funcionamento, com especial ênfase para a sua arquitectura, tecnologias usadas para a interacção com o utilizador e plataformas que compõem a lógica de negócio.

A Universidade de Leeds, por exemplo, implementou um sistema de experimentação remota com a arquitectura ilustrada na Figura 2.

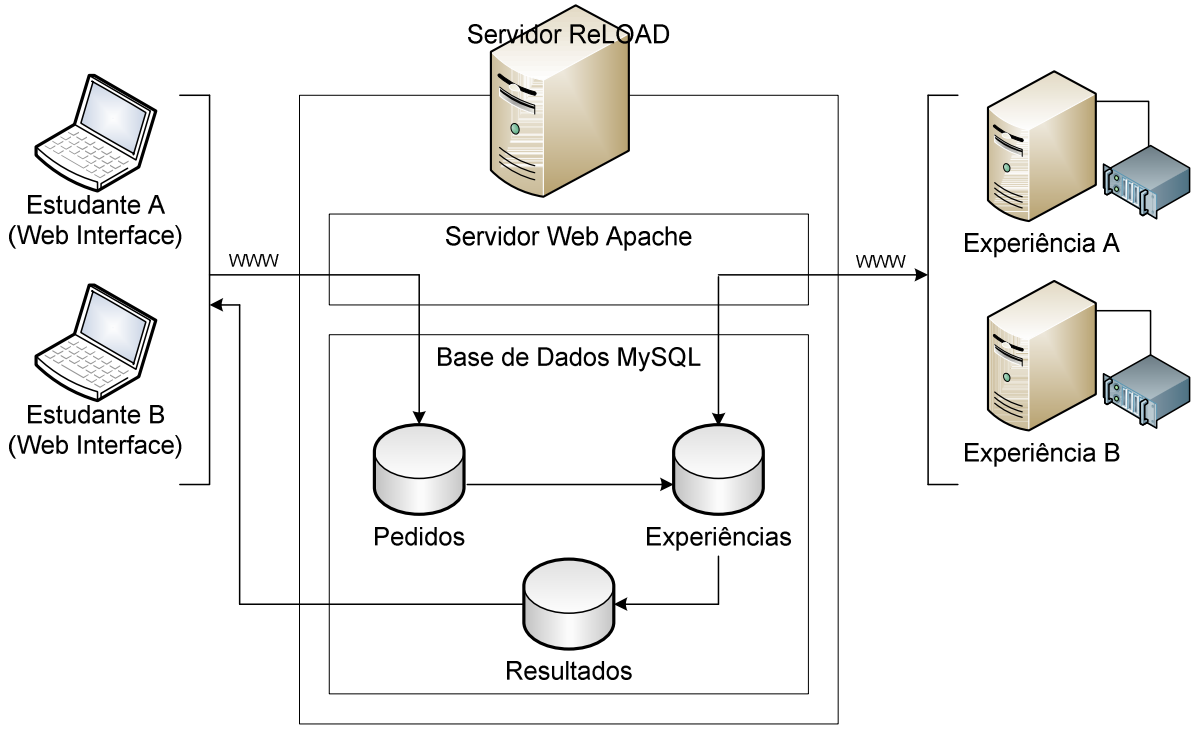

Figura 2 - Arquitectura do sistema ReLOAD [Hanson et al., 2009]

Esta solução caracteriza-se por possuir uma arquitectura distribuída com várias camadas. Os alunos, através de terminais ligados à Internet, acedem a um servidor que serve de camada intermédia. O servidor por sua vez tem várias funções, entre as quais, responder aos pedidos e armazenar informação das experiências em bases de dados. Por fim, existem os dispositivos electrónicos onde são feitas as experiências, e que comunicam igualmente com o servidor através da Internet. Cada vez que um aluno utiliza o sistema, depara-se com uma interface onde pode definir os dados de configuração a enviar. Quando envia o pedido, o servidor trata de armazená-lo numa base de dados, para posteriormente ser tratado pela estação. Após a estação tratar do pedido, comunica de novo com o servidor onde são armazenados os resultados, respondendo desta forma ao aluno. Os dados são apresentados em interfaces que primam pela usabilidade, apresentando os dados, por exemplo, em gráficos e outros componentes de visualização.

Segundo estatísticas oficiais da universidade, os alunos confiam plenamente no laboratório remoto, pois 95% afirma acreditar que os dados são reais. Também enalteceram o facto de poderem aceder ao laboratório 24 horas por dia, 7 dias por semana, pois permite-lhes gerir o tempo da maneira que melhor lhes convém. No entanto, revelaram algumas dificuldades técnicas, pois nem sempre é fácil a assimilação de um novo sistema informático, nem ter a sensação que se está na realidade a interagir com um ambiente real [Hanson et al., 2009].

A Universidade de Bordéus optou por outra plataforma, o eLab. Esta é uma plataforma flexível que providencia acesso a um vasto número de dispositivos, oferecendo diferentes tipos de cenários, o que faz com que a sua aceitação no mundo académico seja ampla. Tem a vantagem de não requerer qualquer *software* comercial, o que diminui substancialmente os custos desta plataforma. É baseado em vários projectos, como o RETWINE (Remote Worldwide Instrumentation, cuja ideia era a partilha dos instrumentos laboratoriais através da Internet) e e-Merge (cujo objectivo é a investigação de uma rede educacional avançada e inovadora para disseminar os laboratórios *online* que suportam as ciências electrotécnicas). Várias Universidades aderiram ao eLab, principalmente as localizadas na Tunísia, Irlanda, Finlândia, Bélgica ou França.

A arquitectura do laboratório remoto implementado na Universidade de Bordéus é ilustrada na Figura 3.

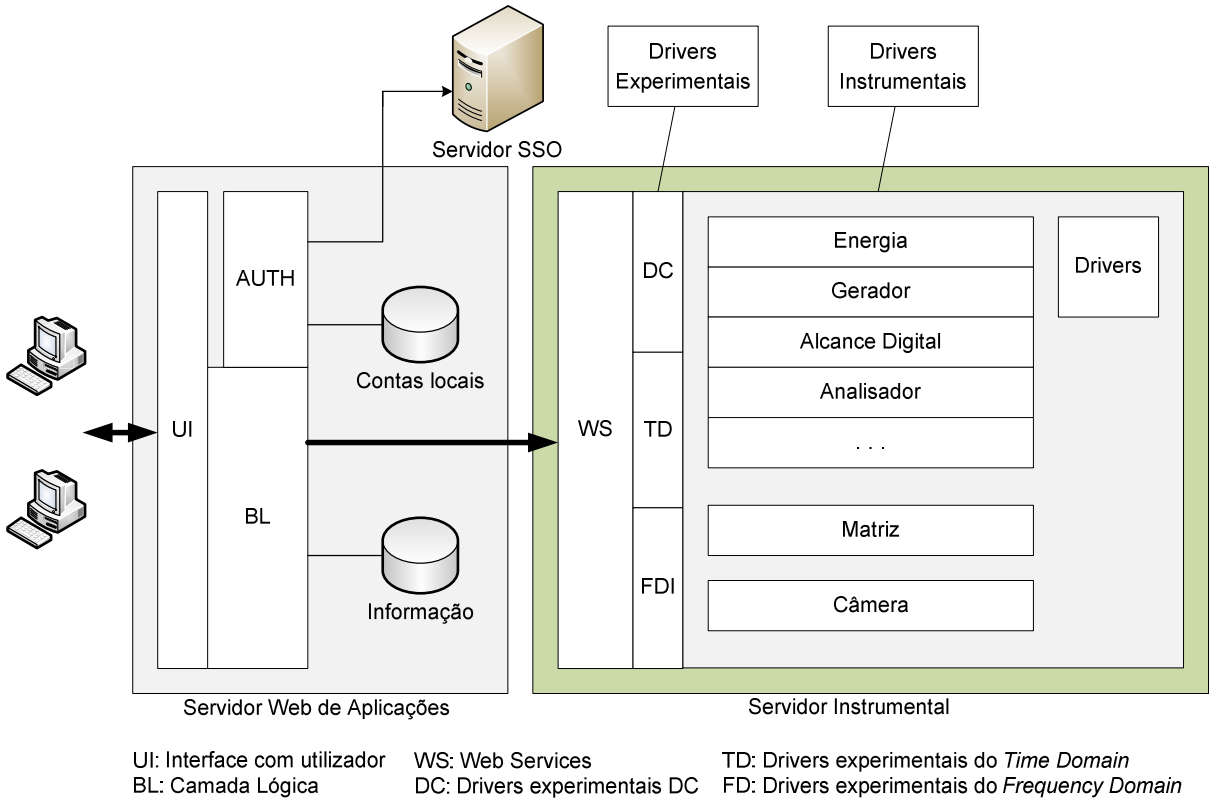

Figura 3 - Arquitectura da plataforma eLab [Lewis et al., 2009]

Pode-se observar que possui uma arquitectura bastante semelhante à descrita anteriormente, para o caso do sistema ReLOAD. No entanto, é de destacar o uso de *Web Services* (SOAP) na comunicação entre a camada intermédia e o servidor instrumental. Este, por sua vez, usa o tradicional "Linux, Apache, MySQL, PHP" (LAMP) para implementar a lógica de negócio. No que diz respeito a pormenores técnicos funcionais, destaca-se a implementação de um *Single Sign On* (SSO), permitindo integrar a autenticação no

laboratório remoto através da plataforma educacional já existente. Este sistema requer que o aluno interaja frequentemente com o sistema, sob pena de ter de se autenticar de novo [Lewis et al., 2009].

Na Universidade de Pisa foi criada uma plataforma, o Jehuty, que abstrai o cliente da comunicação com objectos remotos. É usada para que os alunos tenham acesso a um laboratório remoto, de modo a que possam interagir com objectos remotos que possuem material didáctico. O Jehuty é um servidor capaz de gerir utilizadores, acessos, permissões e agendamentos. A comunicação entre o cliente e o servidor é efectuada recorrendo a *Web Services*. É um sistema modular pois constrói as páginas dinamicamente conforme a informação do utilizador e os dados da base de dados, baseando a sua construção em *Java Servlets*. A arquitectura do sistema está dividida em cinco camadas:

- JXMLDC (*Jehuty XML Database Connector*) – Mecanismo que lê o ficheiro de configuração do sistema;
- JDD (*Jehuty Database Driver*) – Camada que liga à base de dados MySQL;
- JC (*Jehuty Core*) – Usa as duas camadas anteriores para gerir as regras de acesso a cada módulo e verificar as permissões dos utilizadores;
- JM (*Jehuty Modules*) – Inicializados pelo JC, permitem interagir com o sistema;
- JTS (*Jehuty Template System*) – Sistema de modelos que permite separar a camada lógica da camada de apresentação.

No que diz respeito ao sistema de permissões, o sistema suporta quatro tipos de utilizador:

- Administradores de Sistemas – Que adicionam e removem professores;
- Programadores – Para manter o código fonte da aplicação;
- Professores – Adicionam material didáctico no servidor e validam os registos dos alunos;
- Alunos – Após terem feito o registo e serem validados, acedem ao material colocado pelos professores e interagem com os objectos remotos.

A interface gráfica para controlar os objectos remotos foi implementada com recurso a *Applets*, usando a linguagem de programação Java. Também possui um mecanismo de *drag&drop*, capaz de mudar o posicionamento do material didáctico presente no objecto remoto [Balestrino et al., 2009].

Outro exemplo de um sistema semelhante ao anterior é a aplicação JIL (Java-Internet-LabVIEW), que foi desenvolvida na *Universidad Nacional de Educación a Distancia*, em Espanha. Basicamente foi elaborada uma aplicação em Java que corre do lado do servidor, e que é capaz de controlar remotamente um laboratório remoto que está implementado sobre a plataforma LabVIEW. O sistema recorre aos *sockets* TCP para efectuar a ligação com o LabVIEW, embora esteja prevista o suporte a outros protocolos, como o UDP, HTTP e HTTPS. Também possui a particularidade de conseguir trocar imagens e vídeos. Na Figura 4 pode-se observar a arquitectura do JIL. [Dormido et al., 2009]

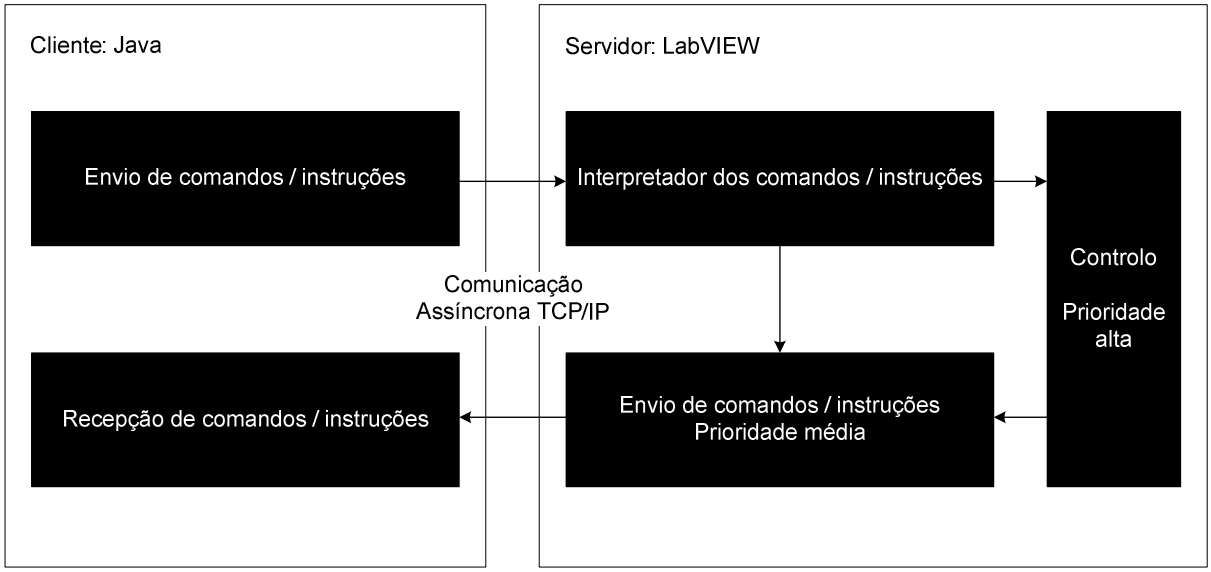

Figura 4 - Arquitectura do sistema JIL

O JIL é capaz de interagir com o utilizador através de uma interface Web, permitindo-o efectuar um conjunto de acções, tais como:

- Iniciar a execução ou parar um VI (cujo significado é explicado na subsecção a seguir) que esteja no laboratório;
- Definir variáveis que estejam no VI;
- Publicar o VI;
- Monitorizar o VI;
- Modificar a conexão da applet de modo a conseguir ligar-se ao VI certo.

Todos os exemplos mencionados foram bastante úteis na medida em que permitiram identificar quais as características e lacunas de laboratórios remotos já em funcionamento real. Todas as características desses sistemas foram consideradas na construção do novo laboratório.

2.2. Plataformas de experimentação remota

O LabVIEW é uma plataforma de programação gráfica, criada pela National Instruments (NI) em 1986, usada por um vasto número de cientistas e engenheiros com a finalidade de desenvolver, testar, manusear e controlar sistemas através de objectos gráficos e fluxos de dados [National Instruments, 2010g]. Caracteriza-se por ser multiplataforma, oferecendo centenas de bibliotecas para análise, captura e visualização de dados. Possui uma comunidade bastante activa, com cerca de cem mil participantes anuais nos grupos de apoio à plataforma. Para além disso, possui suporte personalizado a grupos de utilizadores e parceiros [National Instruments, 2010d]. O LabVIEW possui ainda um vasto conjunto de ferramentas para a aquisição, análise, visualização e armazenamento de dados [National Instruments, 2010].

Os programas elaborados em LabVIEW são chamados de instrumentos virtuais, ou VIs, pois a sua aparência imita na perfeição os instrumentos físicos tradicionais, como os osciloscópios ou multímetros. Os VIs são compostos por três componentes:

- *Front panel*, que é a visualização real da aparência visual do VI;
- *Block diagram*, onde se pode implementar a lógica de programação através de objectos visuais e fluxos de dados;
- Ícon e painel de controlo, usados muitas vezes para definir algumas configurações de interligação do VI com outras estruturas.

Tal como a maioria das linguagens de programação, existe o suporte a estruturas para comunicação sobre rede (*sockets*), tipos de dados, estruturas de dados, ciclos, ficheiros, entre outros. É possível construir uma aplicação modular em LabVIEW, interligando vários VIs através de bibliotecas e outras estruturas.

Na Figura 5 pode-se observar o *front panel* e o *block diagram* de uma aplicação feita em LabVIEW, que está legendada. Os componentes mais importantes, que são usados na maioria das aplicações, são o painel de ferramentas (1), algumas *labels* (2 e 4), controlos numéricos no *front panel* (3), controlo numérico no *block diagram* (5), constante numérica (7), função de multiplicação (8), estrutura de ciclo (17), VI incluído em outro VI (16), gráfico (12), entre outros [Connections, 2010].

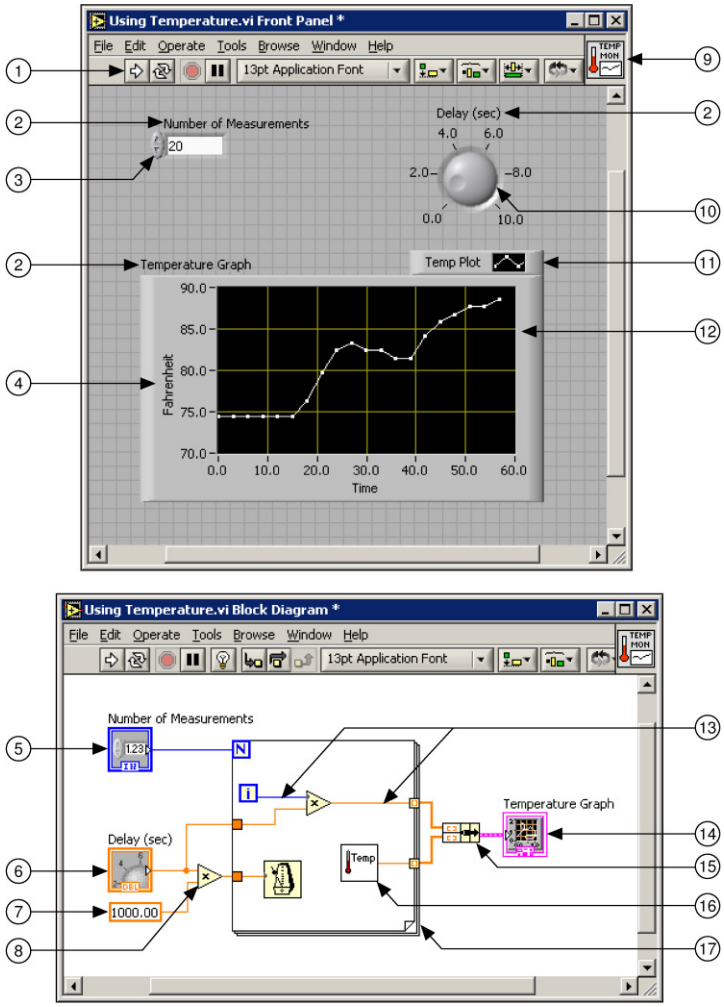

Figura 5 - *Front Panel* e *Block Diagram* de uma aplicação em LabVIEW

Uma das principais áreas de actuação do LabVIEW, e talvez a que maior impulso deu à plataforma, é a aquisição de dados (DAQ) [National Instruments, 2010f]. Para usar esta funcionalidade é necessário que o LabVIEW tenha esse módulo instalado, para que possa oferecer estruturas de acesso e aquisição de dados dos dispositivos electrónicos. Por outro lado, actualmente, o LabVIEW oferece um ambiente capaz de responder às necessidades em várias áreas de actuação, como, por exemplo, área *mobile*, desenvolvimento de sistemas, aplicações em tempo real, construção de módulos FPGA, aplicações embebidas, robótica, redes de sensores *wireless*, entre outros [National Instruments, 2010c]. A disponibilidade destes módulos depende da versão do LabVIEW, que actualmente se divide em Base, Full e Professional [National Instruments, 2010b].

A plataforma LabVIEW possui um mecanismo que permite o acesso remoto a VIs a partir do *Web Browser*. No entanto, é necessário instalar um *plug-in* proprietário que varia conforme o *Web Browser*, a sua versão e o sistema operativo.

Um dos dispositivos físicos mais usados no ambiente académico juntamente com o LabVIEW é a estação NI-ELVIS, ilustrada na Figura 6. Esta estação caracteriza-se por permitir a montagem de circuitos eléctricos numa base do tipo *bread-board*, possibilitando a regulação das entradas e a obtenção dos valores resultantes dos circuitos. Este dispositivo é ligado a um computador, sendo necessária a instalação dos respectivos *drivers* para efectuar a ligação entre o computador e a estação.

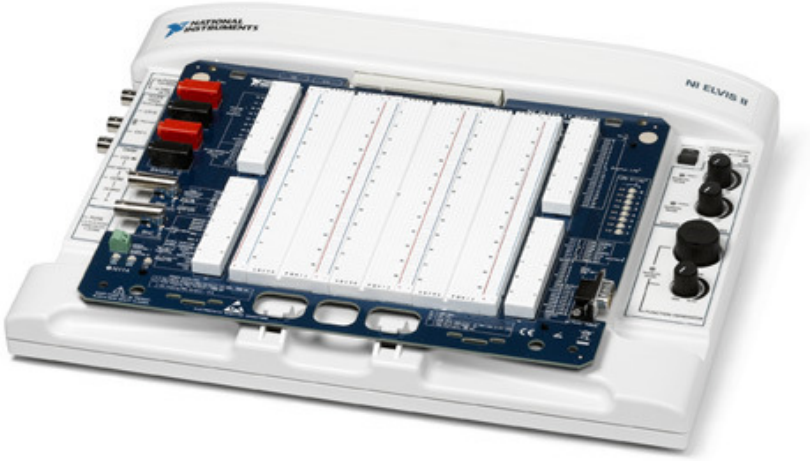

Figura 6 - Estação NI-ELVIS

A estação NI-ELVIS pode ser usada para interagir com várias aplicações, entre as quais se destacam o desenho de circuitos analógicos, desenho de circuitos digitais, instrumentação, sistemas embebidos e telecomunicações.

Como já foi referido anteriormente, com o módulo DAQ do LabVIEW é possível interagir com sistemas para a aquisição e tratamento de sinais. É aqui que surge a necessidade da estação NI-ELVIS, pois esta funciona como suporte físico para essa operação. Através do uso das ferramentas apropriadas é possível construir aplicações em LabVIEW que interagem com a estação [National Instruments, 2010e].

2.3. Tecnologias de construção de interfaces gráficas

A arquitectura das soluções apresentadas na secção anterior exige a existência de uma camada para interagir com o utilizador, normalmente baseada na Web. Por isso, é necessário optar por tecnologias que ofereçam uma grande variedade de recursos e que estes respondam totalmente aos requisitos de um sistema laboratorial. Para além disso, seria muito benéfico, senão obrigatório, optar por tecnologias que cheguem à maior quota de utilizadores possível. Aliando essa necessidade à plataforma de construção de *software*, torna-se viável a opção por tecnologias denominadas de *Rich Internet Applications* (RIA).

O termo "RIA" foi introduzido pela Macromedia em 2002 e suscitou desde logo a aceitação do mercado, de tal forma que os próprios concorrentes da agora Adobe também se referem a este termo. Basicamente as RIA são aplicações Web que se assemelham às tradicionais aplicações para *Desktop* pela capacidade da tecnologia, mas ao invés destas funcionam num *Web Browser*, estando normalmente disponíveis em qualquer lugar, desde que os *Web Browsers* em causa tenham instalado um *plug-in*, que não é mais do que uma máquina virtual ou *framework* capaz de fornecer mecanismos para interpretar e executar as aplicações. As RIA predominam no mercado dos jogos *on-line* e da captura e distribuição de vídeo e áudio. A maioria das aplicações RIA caracteriza-se também por possuir a lógica de negócio numa outra camada, o *middleware*, deixando apenas para a interface a tarefa de interacção com os dados e com o utilizador. Apesar disso, a maioria das RIA não são consideradas padrões da Web, facto que para algumas delas é irrelevante dado a quota de mercado que possuem [Ghoda, 2009].

Existem bastantes *frameworks* de apoio à construção de RIA, entre elas Open Laszlo, Google Web Toolkit e algumas implementações de AJAX/JavaScript como é o caso do JQuery. No entanto, existem três *frameworks* que se distinguem das demais devido à sua aceitação por parte dos utilizadores: Flash/Flex, Java/JavaFX e Microsoft Silverlight. As *frameworks* referidas, excepto o Silverlight, são implementações *open source*, facilitando desta forma possíveis modificações no corpo das *frameworks* ou até a construção de novos componentes baseados nos já existentes.

De acordo com [StatOWL, 2010], o suporte a Flash está presente na maior parte dos computadores de todo o mundo. A sua cota de penetração é de 96,5%. Por seu lado, o suporte a Java encontra-se ligeiramente abaixo, mais concretamente nos 79,7%. Por último, o Microsoft Silverlight possui uma quota bastante distante dos outros dois concorrentes, situando-se nos 47,7%.

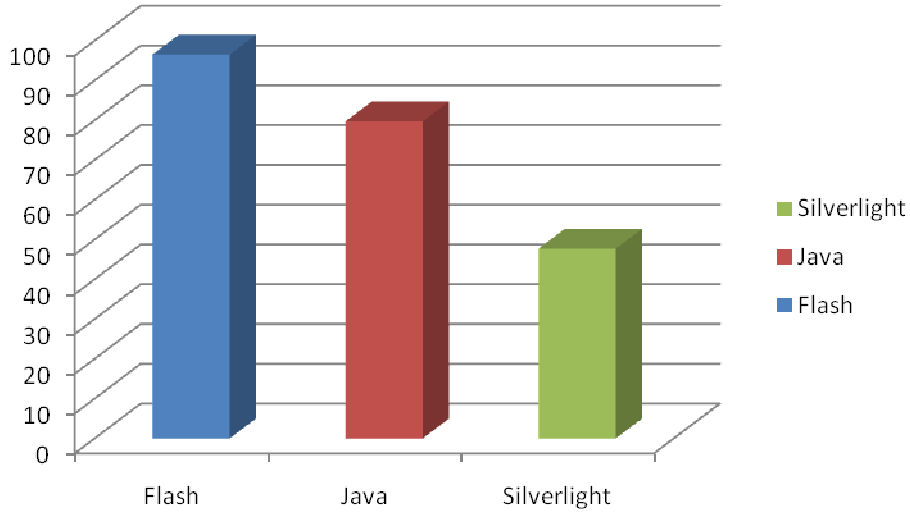

Figura 7 - Quota de mercado das plataformas Flash, Java e Silverlight

Por seu lado, e de acordo com [RIASTATS, 2010], o *plug-in* do Flash é aquele que mais frequentemente se encontra actualizado no sistema dos utilizadores.

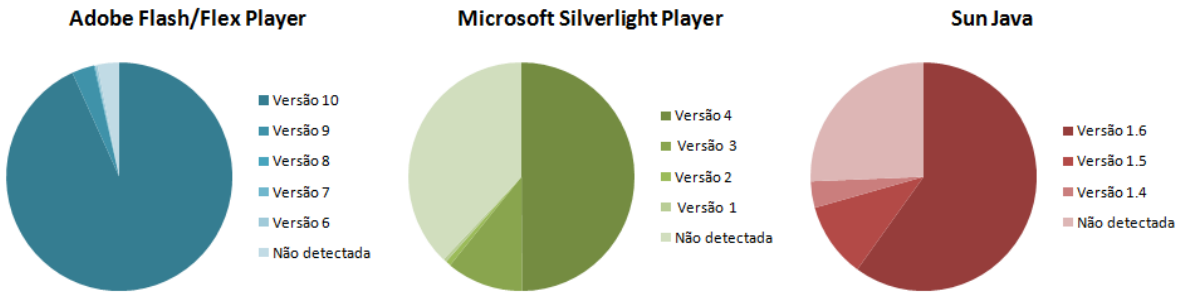

Figura 8 - Versão do *plug-in* instalado por cada plataforma

2.3.1. JavaFX

O JavaFX SDK possui um conjunto de tecnologias, ferramentas e recursos capazes de serem usados pelos programadores para criar interfaces gráficas expressivas e poderosas.

Possui a seu favor o facto de usar uma sintaxe parecida com a linguagem *standard* da plataforma, o Java. Este facto torna-o numa plataforma bastante portável, correndo na maioria dos sistemas operativos e *Web Browsers* existentes no mundo desde que tenha o *plug-in* instalado no sistema. O facto de usar a máquina virtual do Java SE, torna-o bastante estável e com uma enorme capacidade de actuação.

A nível de desenvolvimento pode-se ressaltar dois poderosos ambientes de desenvolvimento, o Netbeans e o Eclipse. Ambos possuem ferramentas que ajudam o programador na construção das aplicações em JavaFX. O Netbeans destaca-se por oferecer uma interface com possibilidade *drag&drop* de componentes visuais [JavaFX Tools, 2010].

A arquitectura da plataforma [JavaFX Platform, 2010] está especificada na Figura 9, saltando à vista desde já o facto de correr sobre a JVM.

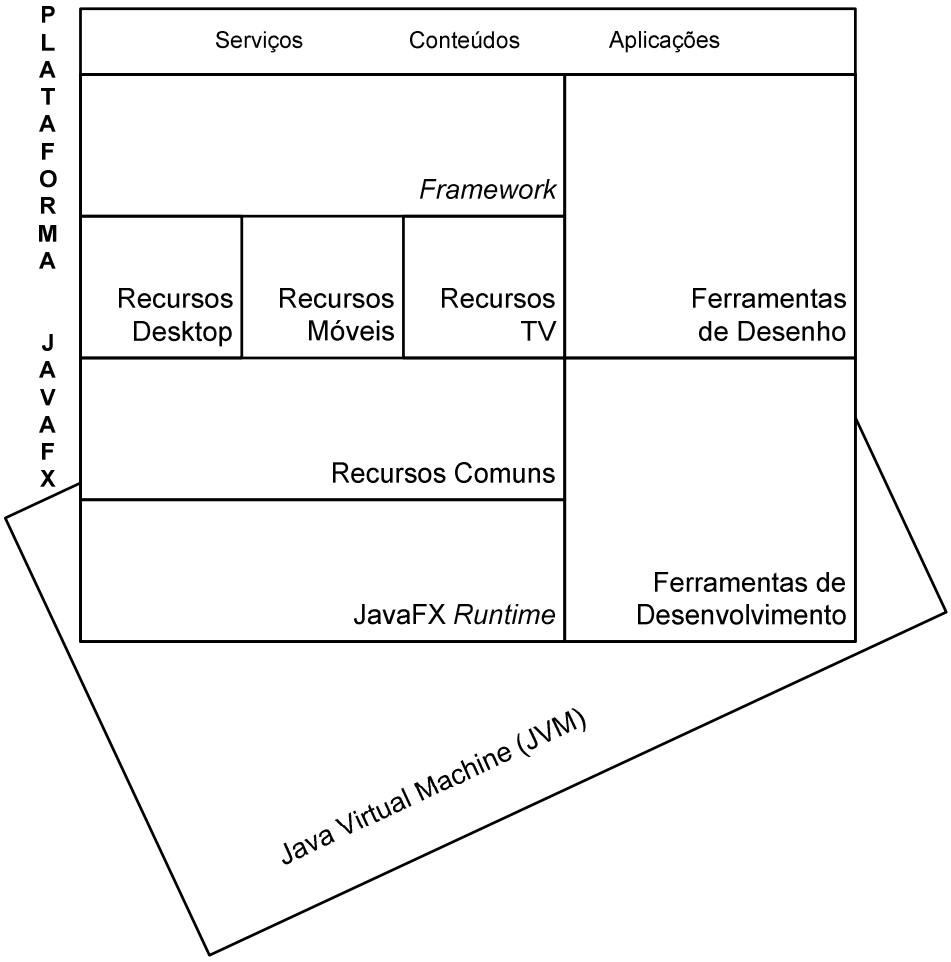

Figura 9 - Arquitectura geral do JavaFX

Como se pode observar na Figura 9, é possível uma aplicação feita em JavaFX correr tanto na Web, como em *Desktop*, televisões ou até *mobile*, desde que o sistema tenha a VM do Java instalado. A plataforma está dividida em duas grandes áreas: uma que oferece ferramentas de desenho de aplicações para os vários ambientes (televisão, móvel ou *Desktop*) e a outra que oferece ferramentas para os programadores desenvolverem a lógica das aplicações [JavaFX, 2010].

O *plug-in* necessário para correr as aplicações desenvolvidas em JavaFX é o Java Runtime Edition (JRE), o mesmo para as restantes aplicações que usam o Java *standard*. O instalador deste *plug-in* ocupa cerca de 16Mb no sistema operativo Microsoft Windows e 20Mb em ambientes Linux.

Apesar de ser anunciado como plataforma *mobile*, este mesmo facto não se reflecte na realidade, pois é escassa a utilização a este nível [Weaver et al., 2009].

2.3.2. Microsoft Silverlight

Microsoft Silverlight é uma tecnologia desenvolvida pela Microsoft com o intuito de competir com o Adobe Flex. Foi lançada em 2007 e possui a particularidade de separar o ciclo de desenvolvimento entre designers e programadores através de uma separação clara do que é interface e do que é funcionalidade na sua arquitectura. A tecnologia foi idealizada para que funcione em diversas plataformas e *Web Browsers*, mas na prática só funciona completamente em plataformas que possuem o sistema operativo Microsoft Windows. Existe também uma implementação *open source* desta plataforma, o Moonlight do projecto Mono que é suportado pela Novell e Microsoft. No entanto, o seu desenvolvimento depende sempre da versão proprietária, o que faz com que seja lançado sempre com alguns meses de intervalo [Silverlight, 2010].

A plataforma do Silverlight usa um *codec* proprietário para transmissão de vídeos, optimizando desta forma o seu desempenho, pois é possível uma transferência mais ampla dos dados e a visualização dos filmes em alta definição. Também suporta aceleração 3D e áudio.

A arquitectura de uma aplicação é dividida em duas partes: os arquivos XAML, (que contêm a linguagem Silverlight propriamente dita); e os arquivos de código com a lógica de negócio, que pode ser C#, Visual Basic, Python ou Ruby. Na Figura 10 está presente a arquitectura do Silverlight. À primeira vista ressalta o facto de possuir duas partes, interligadas por XAML: a primeira parte é constituída por recursos que facilitam o acesso aos dados, comunicação entre módulos e implementação de lógica aplicacional; a segunda parte foca-se mais na disponibilização de recursos para o tratamento de vídeo e da própria interface [MSDN, 2010].

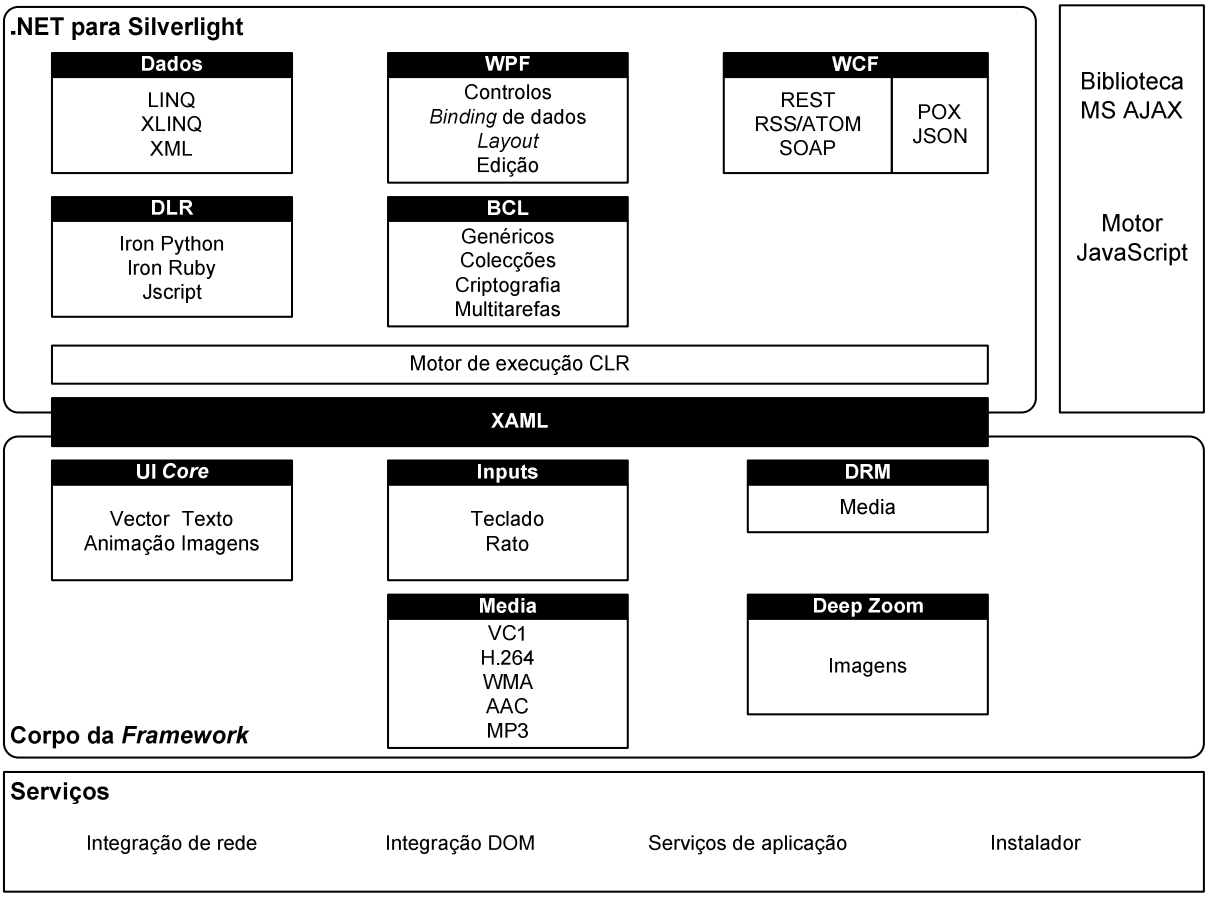

Figura 10 - Arquitectura geral do Silverlight [MSDN, 2010]

O *plug-in* ocupa cerca de 6Mb, funcionando tanto em ambiente Web como em *Desktop*, e até em *mobile* [Microsoft Silverlight, 2010b], muito embora a utilização seja escassa em ambientes fora do *Desktop* [Microsoft Silverlight, 2010].

2.3.3. Adobe Flex

Adobe Flex é uma *framework* desenvolvida pela Adobe Systems para o desenvolvimento de aplicações para Internet baseadas em Flash, capazes de correr em várias plataformas e sistemas operativos. Foi inicialmente lançada em 2004 pela Macromedia, e numa primeira versão não teve a aceitação do mercado desejada, pois as licenças eram vendidas por cada servidor que suportava a tecnologia. Posteriormente, e com a compra da Macromedia por parte da Adobe, foram lançadas outras versões da *framework*, sempre com novas capacidades para a construção de aplicações [Adobe, 2010b]. A última versão da *framework* foi um grande avanço em relação às outras versões, no que diz respeito à arquitectura, pois permite separar a camada visual, com recursos de construção da interface, da camada lógica, que disponibiliza recursos de interligação dos componentes visuais com as restantes estruturas [Giametta, 2009].

Uma aplicação em Flex é dividida em 2 partes: o código MXML, que define a interface em si; e o código Action Script 3, que permite implementar a lógica inerente à interface implementada. No entanto, é possível colocar código Action Script 3 em MXML, ou então optar por elaborar toda a aplicação, ou pequenas partes, em Action Script 3, em detrimento do MXML, pois o MXML não é mais do que uma camada de abstracção que ajuda na construção e uso de componentes elaborados em Action Script 3 [Brown, 2007]. A Figura 11 representa as grandes áreas da arquitectura da *framework* do Flex. Como se pode observar, as camadas MXML e Action Script servem para especificar as aplicações, recorrendo à biblioteca de classes do SDK do Flex. A comunicação com outras camadas é feita através de serviços de acesso e comunicação, especificados em protocolos (SOAP, REST, AMFPHP, BlazeDS).

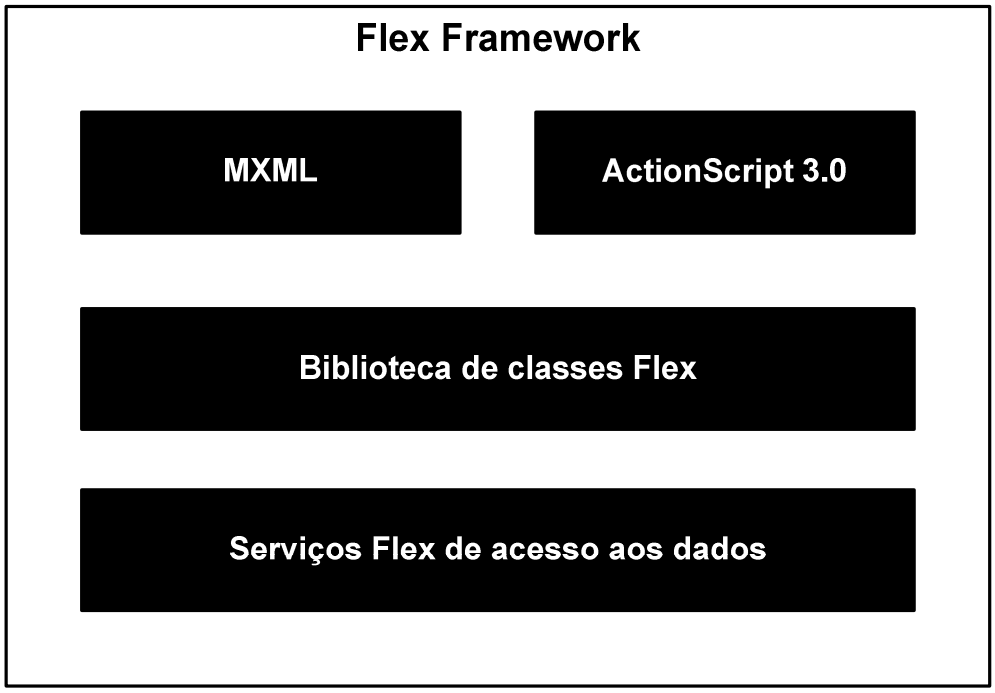

Figura 11 - Arquitectura geral da Flex Framework [Brown, 2007]

A nível de IDEs, pode-se usar o Flex Builder, que é uma implementação do IDE Eclipse com funcionalidades para desenvolver em Flex. Para além disso, é possível usar o Eclipse tradicional com o *plug-in* do Flex Builder, facilitando assim a vida aos programadores de outras linguagens ao permitir ter apenas um único IDE aberto. As licenças do Flex Builder são comerciais e ajudam os programadores pelo facto de ser possível o *drag&drop* de componentes para as interfaces, mas a Adobe disponibiliza um compilador de forma gratuita para quem não quiser ou não puder comprar o Flex Builder, o mxmlc [Gassner, 2010].

As aplicações elaboradas em Flex possuem normalmente uma outra camada lógica que corre no servidor, e que serve para aceder à base de dados e/ou implementar a lógica de negócio. Normalmente são usadas linguagens como C#, Java, PHP, Ruby ou até Coldfusion. Essa camada também tem a função de interagir com a interface em Flex, podendo para isso ser usadas várias formas de comunicação, desde simples pedidos http, uso de *Web Services* ou através de *remoting* (*Remote Object*). Esta última tecnologia permite o mapeamento real de um objecto presente no servidor para outro presente no cliente, apoiado por uma *framework* de comunicação que, no caso das linguagens referidas, pode ser o BlazeDS para Java ou o AMFPHP para PHP. Estes protocolos são binários e, por isso, têm um excelente desempenho a nível de transferência de dados [Adobe, 2010].

O *plug-in* do Flash, necessário para correr as aplicações em Flex, ocupa cerca de 1,8Mb, estando presente na quase totalidade dos computadores mundiais (97%), gozando do facto de ser multiplataforma. Usando a plataforma Flex é possível desenvolver aplicações para Web, *Desktop* (Adobe AIR) e muito recentemente para *mobile*.

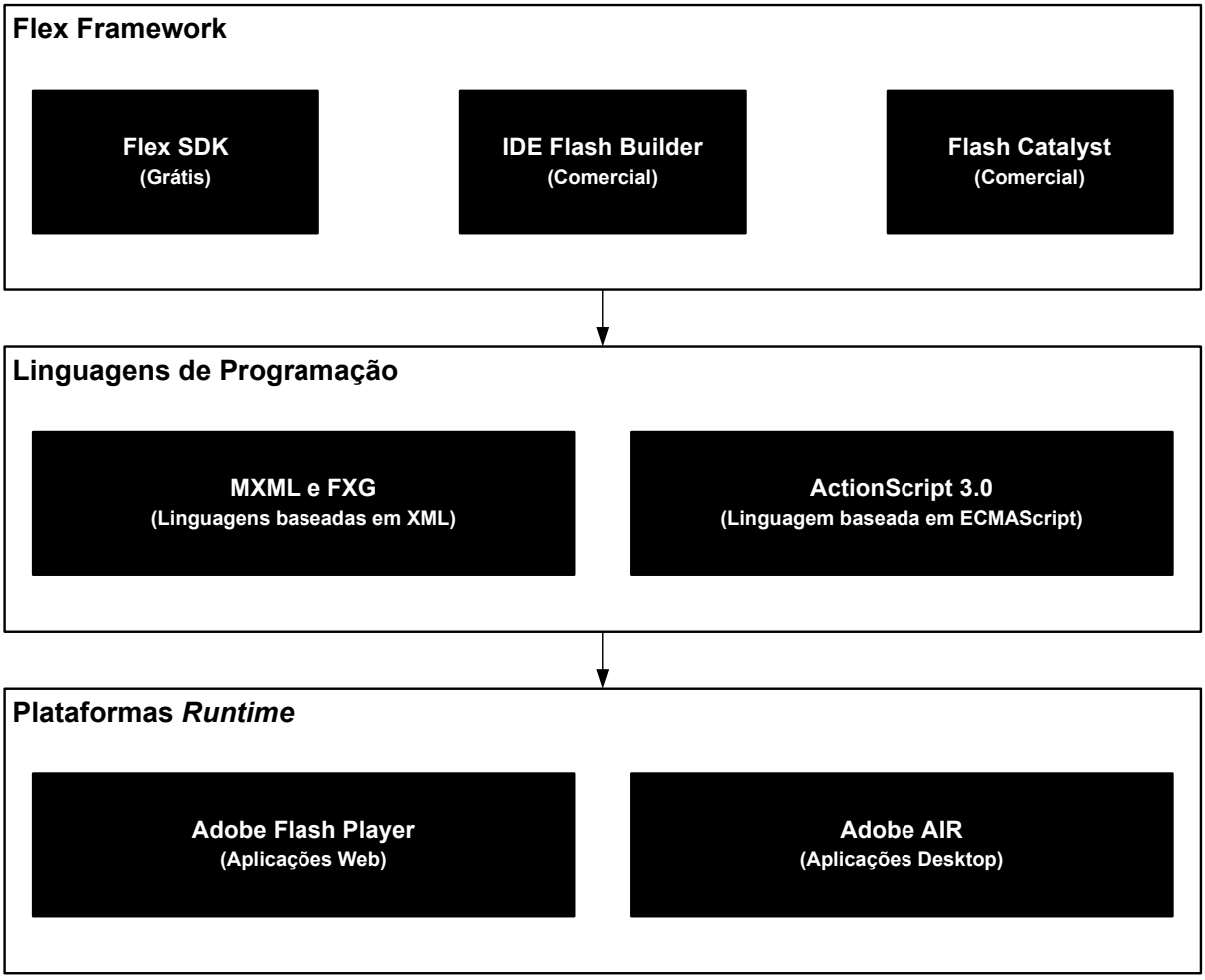

Figura 12 - Visão geral da plataforma Flex [Gassner, 2010]

Também é possível criar interfaces visuais com uma ferramenta da Adobe, mais concretamente o Adobe Catalyst. Esta ferramenta possui mecanismos de exportação dessas interfaces para MXML e consequentemente para Action Script. Isto só é possível desde o lançamento da versão 4 do Adobe Flex [Gassner, 2010]. A Figura 12 especifica o que foi descrito anteriormente, mostrando que são necessárias as ferramentas de desenvolvimento para que, com as linguagens de programação, seja possível construir aplicações que corram em sistemas que tenham o *plug-in* do Flash instalado [Adobe, 2010c].

Neste capítulo fez-se uma síntese do estado da arte no que se refere aos laboratórios remotos e às tecnologias, a nível das aplicações de *software*, que permitem a interacção entre o utilizador e os laboratórios remotos. No próximo capítulo, será descrita a forma como estas tecnologias foram usadas no caso concreto do laboratório remoto para o apoio ao ensino da electrónica para melhorar a disponibilidade e portabilidade da sua utilização.

3. Da identificação de requisitos às opções de implementação

3.1. A não universalidade da solução actual

O Laboratório de Investigação em Sistemas de Teste (LABORIS) é um centro de investigação que surgiu durante o ano 2002, estando actualmente sediado nas instalações do ISEP. Tem como objectivo principal desenvolver soluções de teste e depuração que respondam às necessidades dos sistemas e tecnologias actuais, apoiando também as actividades académicas e a formação de alunos do ISEP, estabelecendo parcerias com outros grupos de investigação da mesma área [LABORIS, 2010].

Nesse sentido, e tendo em conta o que foi descrito ao longo do capítulo 2, o LABORIS possui um laboratório remoto que permite aos alunos da disciplina de Electrónica aceder remotamente a experiências remotas. O laboratório foi desenvolvido no âmbito de um trabalho de Mestrado que decorreu no LABORIS [Sousa, 2008], e tem por base uma estação NI-ELVIS gerida a partir de uma aplicação em LabVIEW. A aplicação permite que o utilizador defina, controle e observe circuitos eléctricos e electrónicos presentes na estação NI-ELVIS, a qual se encontra ligada a um computador, o servidor de experiências, que implementa a interface remota entre o utilizador e a experiência.

Para estabelecer a ligação ao servidor de experiências foi elaborado, com recurso à linguagem PHP, um módulo para o Moodle que comunica com os VIs da aplicação através de *sockets*. Os VIs estão à escuta de pedidos por parte do módulo, e têm a função de informar se é possível realizar a experiência ou não. Falta referir a forma como o utilizador interage com a aplicação. Depois de autenticar-se no Moodle e aceder ao módulo desenvolvido, é efectuada uma comunicação através de *sockets* com os VIs para perceber se o laboratório está disponível para ser acedido. Em caso afirmativo é gerado um painel remoto com a interface do VI, é enviado para a interface o endereço desse painel remoto, que está embutido num ficheiro html, e é acedido através de um *Web Browser*. O módulo reencaminha o utilizador para essa página gerada (com um endereço único), de modo a que este possa interagir com o laboratório. O *workflow* descrito está representado na Figura 13.

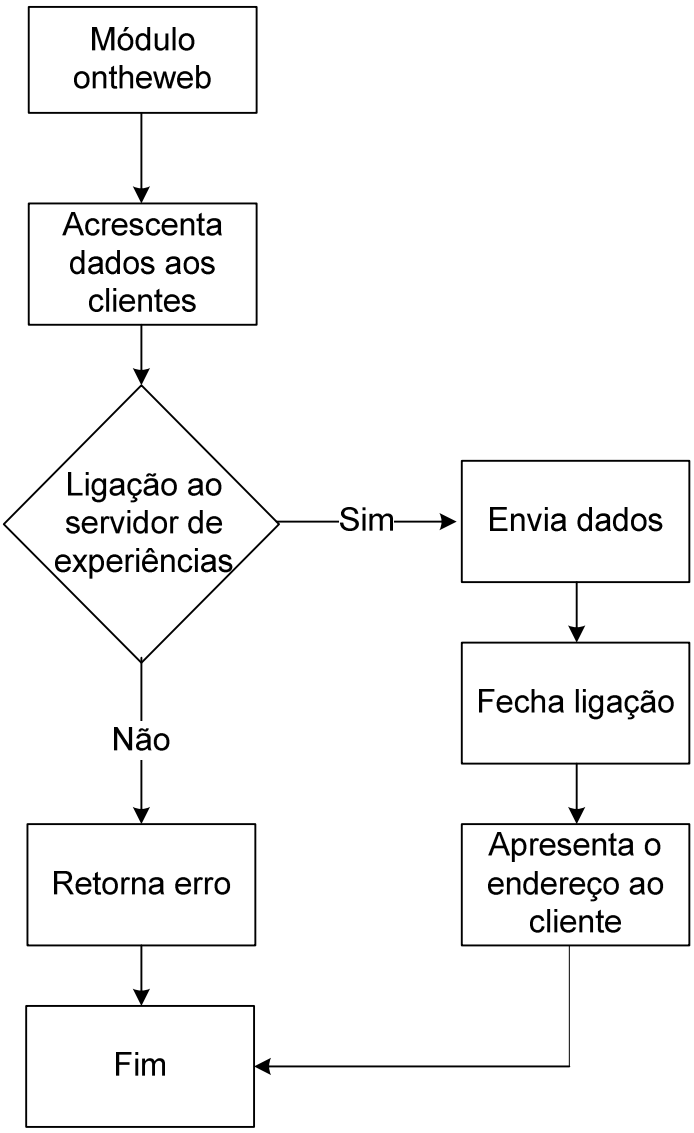

Figura 13 - *Workflow* da solução existente

Tal como descrito anteriormente, a arquitectura do laboratório existente é distribuída por várias camadas. Olhando para a Figura 14, vê-se realçado o nível 1 que é o *user interface* da aplicação, o nível 2 que possui a lógica (servidor que aloja o Moodle) e o nível 3 onde se encontra o servidor de experiências, que possui a estação NI-ELVIS ligada.

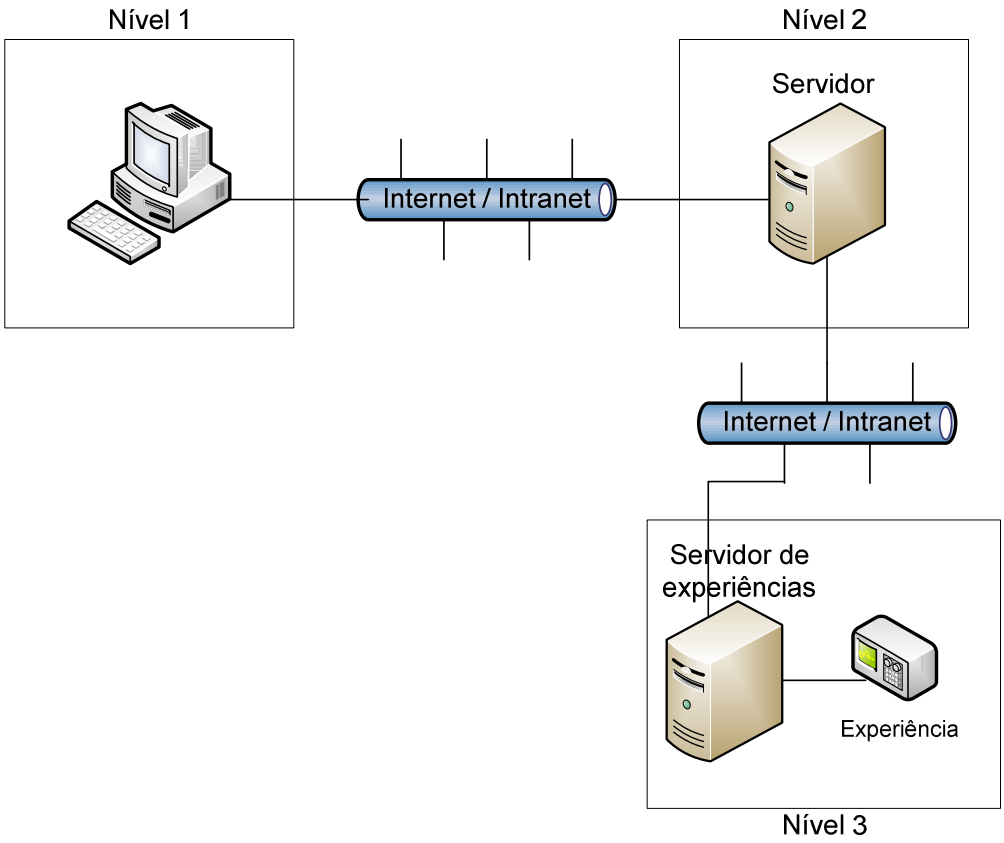

Figura 14 - Arquitectura implementada para o laboratório remoto em funcionamento

A solução explicada anteriormente tem funcionado e cumprido com os requisitos mínimos que uma aplicação deste tipo exige. Contudo, existem vários problemas associados à utilização da solução existente, como sejam:

- Necessidade de instalação de um ***plug-in***: É necessário que os utilizadores instalem um *plug-in* proprietário para conseguirem aceder remotamente ao laboratório. A instalação deste *plug-in* levanta uma série de problemas aos utilizadores, como por exemplo o facto de este ocupar algumas dezenas de Mb e servir apenas para o uso desta aplicação. Para além disso, existe um *plug-in* diferente por cada sistema operativo, *Web Browser* e arquitectura do processador (32 e 64 bits). Apesar disso, e segundo a experiência de uso da solução actual, a aplicação só funciona perfeitamente no sistema operativo Microsoft Windows desde que esteja a ser usada no Internet Explorer;
- **Sobrecarga** do servidor de experiências: O facto de abrir um painel remoto por cada cliente ligado faz com que possa haver vários painéis remotos abertos ao mesmo tempo, aumentando substancialmente a carga do servidor de experiências. Por cada painel remoto aberto é gerado um ficheiro html com um endereço único, tendo a aplicação que controlar esses ficheiros;

- **Tempo de carregamento** da interface: A interface com o VI remoto demora, em média, cerca de 30 segundos a ser carregada;
- Existência de um **tempo máximo** para se realizar a operação: Após aceder à aplicação, o utilizador possui trinta minutos para submeter os dados, sob pena da sessão ser encerrada;
- Problemas com ***firewalls***: O *plug-in* proprietário da NI provoca fluxos de comunicações por várias portas, pelo que, por vezes, não é possível o estabelecimento da comunicação quando a *firewall* da rede local de onde se pretende aceder tem essas portas fechadas, ou quando o gestor da rede do próprio ISEP muda as definições da *firewall*.

Todos estes problemas levaram a que fosse proposta esta dissertação, com o objectivo de conceber, arquitectar e desenvolver uma solução que os permita resolver e ser usada de forma compacta.

3.2. Identificação de requisitos da nova solução

No sentido de responder às necessidades de desenvolvimento de uma solução que resolva os problemas descritos no subcapítulo anterior, foram levantados os seus requisitos funcionais e não funcionais. Em relação aos requisitos funcionais, foram definidos os seguintes:

- A aplicação deve possuir um sistema básico de autenticação de modo a permitir acesso apenas a utilizadores com permissões;
- A aplicação deve estar disponível em dois idiomas diferentes, o português e o inglês;
- O utilizador deve interagir com a aplicação através de uma interface Web de modo a ser possível o acesso a qualquer hora e de qualquer sistema operativo ou *Web Browser*;
- O utilizador deve poder definir todos os valores de configuração (frequência, amplitude, número de amostras, entre outros) que existem na actual aplicação na interface da nova solução;
- O utilizador deve poder enviar um pedido com os valores de configuração definidos para o servidor;

- Deve ser tomado em conta o facto de haver vários utilizadores a aceder ao mesmo tempo à aplicação, tendo de existir uma fila de espera do tipo FIFO para que os pedidos mais antigos sejam tratados antes dos novos pedidos;
- O utilizador deve poder ter acesso aos resultados das requisições enviadas para o servidor quando o pedido já tiver sido tratado;
- O utilizador deve poder cancelar um pedido feito anteriormente;
- O docente deve ser capaz de gerir os acessos ao laboratório remoto;
- O utilizador deve poder sair da aplicação quando assim o desejar.

No que diz respeito aos requisitos não funcionais, devem ser tomados em conta os seguintes pontos:

- A interface com o utilizador deve ser intuitiva e de fácil utilização;
- A aplicação não deve deixar o utilizador inserir valores que não estejam dentro dos limites aceitáveis por cada unidade;
- Os resultados devem ser mostrados, se possível, em componentes de visualização capazes de aumentar a percepção ao utilizador e de fazer comparação, como por exemplo, tabelas e gráficos;
- A aplicação não deve provocar grande carga sobre o servidor de experiências;
- A aplicação deve ser segura, isto é, protegida contra acessos indevidos e impossibilitando que o utilizador aceda a dados aos quais não está autorizado.

Face a estes requisitos, foram identificados pelo menos três tipos de utilizadores: o utilizador normal (aluno), que acede e interage com o laboratório remoto; o regente da disciplina, capaz de permitir o acesso ao laboratório remoto; o administrador do Moodle, com funções de gerir utilizadores e manter a plataforma *on-line*. O *workflow* realizado pelo administrador do sistema é independente da implementação do laboratório, por isso vai ser feita uma focagem nas acções do aluno e docente, como está presente no diagrama de casos de utilização que ilustra a Figura 15.

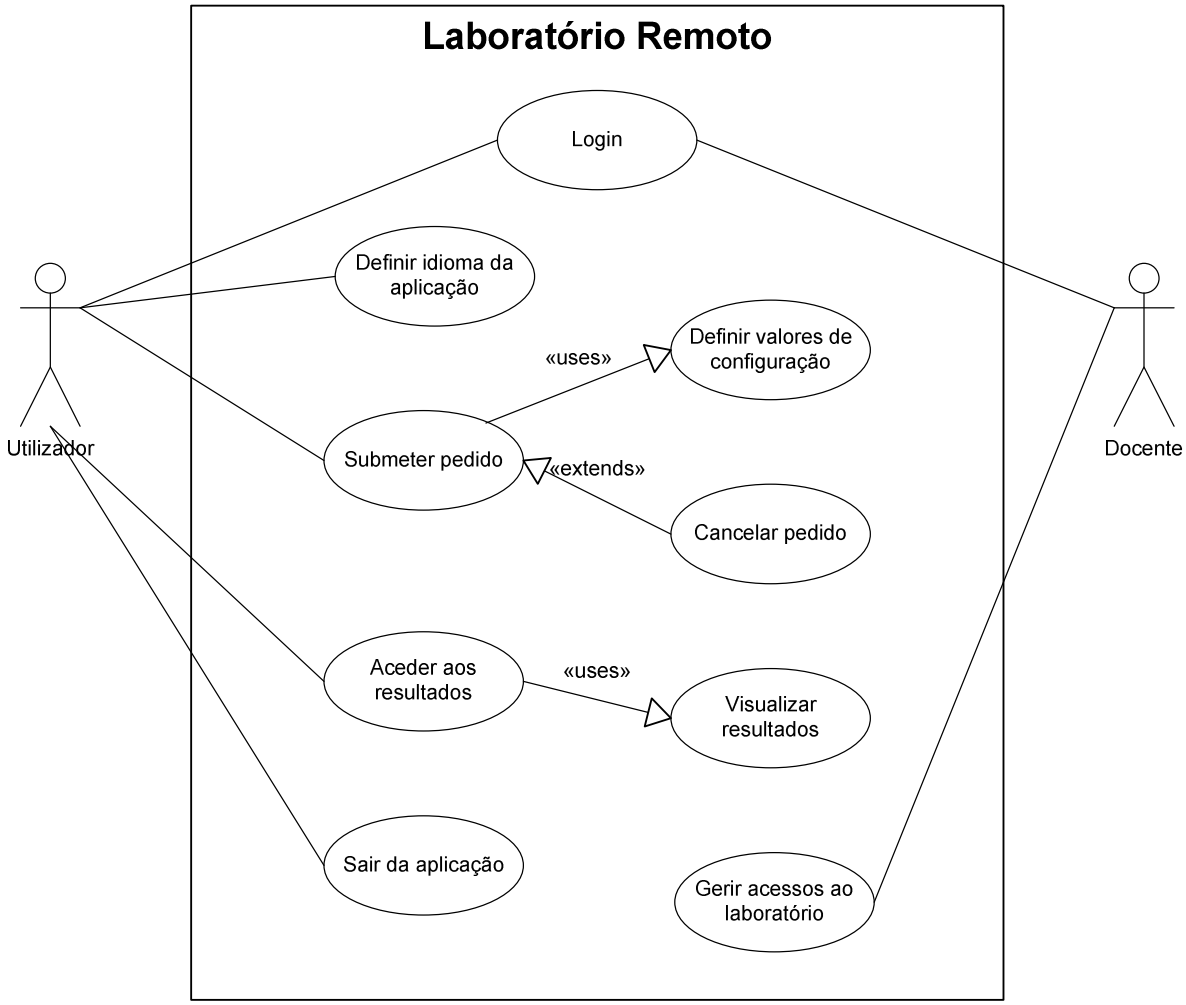

Figura 15 - Diagrama de casos de utilização

No diagrama da Figura 15 é feita a referência ao caso de utilização “Login” e “Sair da aplicação”. No entanto, a implementação das funcionalidades relacionadas com esses casos de utilização vai estar dependente da plataforma Moodle, como vai ser referido num capítulo posterior.

3.3. Soluções técnicas e opções

Antes de iniciar o desenvolvimento do novo laboratório remoto foi necessário tomar algumas decisões suportadas no que foi referido nos anteriores capítulos, de modo a arquitectar uma solução desde a sua base. Neste capítulo serão justificadas as opções das tecnologias mais relevantes.

A avaliar pelas soluções de *software* que são desenvolvidas actualmente e com base na investigação descrita no capítulo 2, é normal existirem várias camadas que dividem a arquitectura de uma aplicação. Na sua maioria, essas aplicações são compostas por três camadas: *front-end*, que serve para o utilizador interagir com a aplicação e que também é

chamada de interface com o utilizador; *middleware*, usada para a implementação da lógica de negócio da aplicação; *back-end*, que trata do acesso e armazenamento dos dados. Nesse sentido a solução do novo laboratório remoto foi projectada de modo a obedecer a esta arquitectura, conforme ilustra a Figura 16.

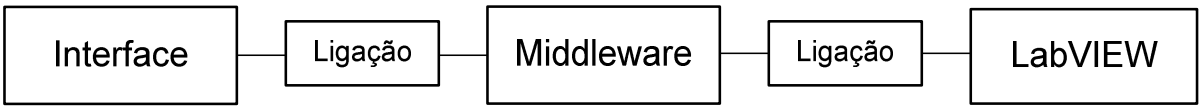

Figura 16 - Arquitectura simplificada do novo laboratório remoto

Como se pode observar, existe uma interface que é manuseada pelo utilizador e comunica com a camada de negócio (*middleware*). Por sua vez a camada de negócio vai comunicar com o LabVIEW. Não está representado o acesso à estação NI-ELVIS, pois o LabVIEW, através do módulo DAQ, vai enviar valores de configuração e receber os respectivos resultados. Nos subcapítulos que se seguem estão justificadas todas as opções tomadas para cada uma destas camadas, inclusive a forma de comunicação entre elas.

3.3.1. Flex como tecnologia para a construção da interface

Esta escolha é a que tem cariz mais importante entre todas, pois vai ser esta camada a responsável pela interacção com o utilizador, logo a que pretende resolver os principais problemas existentes na aplicação actual.

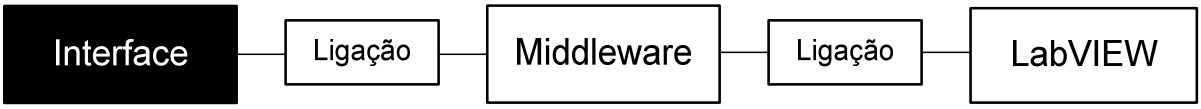

Figura 17 - Arquitectura simplificada do novo laboratório remoto: Interface

Com base na investigação elaborada no capítulo 2.3, chegou-se à conclusão que a tecnologia mais viável para a implementação da interface é o Adobe Flex, pois é superior às outras tecnologias nos seguintes pontos:

- Possui a maior quota de mercado, pois cerca de 97% dos computadores possuem o Flash instalado;
- Possui uma proliferação rápida quando sai uma nova versão do *plug-in*;
- O *plug-in* ocupa menos espaço, com cerca de 1,8Mb;
- É multiplataforma, podendo ser utilizado em vários sistemas operativos (Microsoft Windows, Mac OS e ambientes Linux) e em vários *Web Browsers* (Mozilla Firefox, Internet Explorer, Chrome, Safari, Opera, entre outros);
- É uma tecnologia madura pois a primeira versão foi lançada já em 2004;

- Existem perspectivas animadoras no que diz respeito ao seu uso em dispositivos móveis.

Para além destes pontos resta destacar que o Adobe Flex possui mecanismos de suporte à comunicação usando vários protocolos como *remoting*, *Web Services*, http, entre outros. A comunidade que envolve a tecnologia é bastante activa e possui documentação de boa qualidade, facto que ajuda bastante no decorrer do desenvolvimento de aplicações.

Também houve uma certa ponderação em relação à versão do Flex SDK a usar, pois no decorrer do desenvolvimento da dissertação a Adobe lançou uma nova versão, o Flex 4. No entanto, a versão do Flex 3 foi lançada em princípio de 2008, possuindo assim uma *framework* mais madura e estável pois foram lançadas várias correcções da versão até à saída da versão 4. Por tal, optou-se por utilizar a versão 3 do Flex SDK [Adobe, 2010d].

Em relação ao IDE, foi usado o Flex Builder, que a Adobe disponibiliza gratuitamente para estudantes através do preenchimento de um formulário, o que permitiu obter grandes ganhos de produtividade [Adobe, 2010e].

3.3.2. Versão do LabVIEW

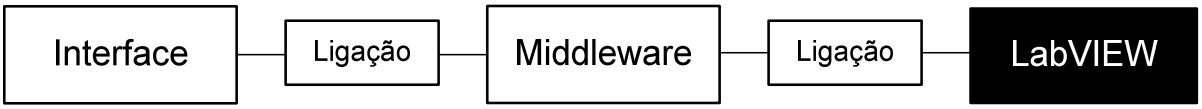

Figura 18 - Arquitectura simplificada do novo laboratório remoto: LabVIEW

O laboratório em funcionamento usava a versão 8.5 do LabVIEW, o qual cobria as necessidades que o sistema exigia, isto é, era capaz de lançar painéis remotos, tinha capacidades de aquisição de dados (DAQ), possuía mecanismos de comunicação de dados através de *sockets* (TCP e UDP) e variáveis partilhadas (mecanismo que permite ler e escrever dados em variáveis que depois podem ser partilhados por toda a aplicação em LabVIEW). Para além disso, à data de implementação do laboratório remoto, a versão 8.5 era a mais recente que a NI tinha lançado.

Após a versão 8.5 a NI lançou actualizações do LabVIEW, e entre elas destaca-se o LabVIEW 2009 que foi um importante passo rumo à modernização que os sistemas actuais exigem. O LabVIEW 2009, para além das funcionalidades que as anteriores versões já disponibilizavam, possui várias melhorias, entre elas destacando-se o suporte a sistemas de 64 bits, a introdução de mecanismos de *debug* avançado, a melhoria no suporte à construção de módulos FPGA, a capacidade de interacção com outros formatos de documentos, os novos componentes 2D e 3D de visualização de dados, uma *framework* de testes unitários, entre outras. No entanto, a melhoria que mais impacto tem para esta

dissertação é o facto de suportar a construção de *Web Services*. Isto é, a possibilidade de os dados poderem ser controlados através de simples pedidos http. Este assunto vai ser desenvolvido no próximo subcapítulo [National Instruments, 2010h].

O facto da versão 2009 do LabVIEW ser compatível com a versão 8.5 também facilita e diminui o esforço quando se faz o *upgrade* da plataforma, permitindo que as anteriores aplicações continuem a funcionar do mesmo modo que funcionavam na anterior versão. Este facto é fundamental para o desenvolvimento do novo laboratório remoto pois permite aproveitar a aplicação que existia no antigo [National Instruments, 2010i].

Tendo em conta o que foi descrito anteriormente, decidiu-se então fazer o *upgrade* da versão 8.5 do LabVIEW para o LabVIEW 2009. Este *upgrade* torna-se muito vantajoso, como vai ser explicado nos subcapítulos seguintes.

3.3.3. Ligação entre *Middleware* e LabVIEW

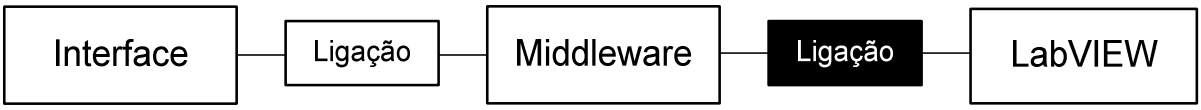

Figura 19 - Arquitectura simplificada do novo laboratório remoto: *Middleware* ↔ LabVIEW

A escolha do método de comunicação entre a camada intermédia e o LabVIEW mereceu uma profunda investigação na medida em que a nova versão do LabVIEW 2009 oferece uma funcionalidade que as anteriores versões não suportavam, os *Web Services*.

O antigo laboratório remoto usava *sockets* como forma de comunicação entre o módulo PHP do Moodle e o LabVIEW. Esta comunicação apenas era usada para fazer um pedido ao servidor de experiências para gerar um painel remoto numa área que aloja os ficheiros do servidor Web, pedido ao qual este respondia com o nome do ficheiro para o utilizador aceder através do navegador. No actual laboratório, seria possível optar por uma solução deste tipo mas implicaria desde já a alteração do VI do painel remoto, de modo a colocar todas as opções de configuração com um *socket*, que estaria à escuta em determinada porta. No entanto, como existem bastantes valores de configuração, isto iria fazer com que fossem necessários vários *sockets*, implicando uma alteração maior no VI. Para além disso, seriam necessárias bastantes portas abertas para o exterior, o que aumentaria ainda mais os problemas já conhecidos com as *firewalls*. Por último, também seria necessário um maior esforço na gestão das ligações do lado da camada intermédia, dada a necessidade de gerir um grande número de ligações activas.

O LabVIEW disponibiliza também uma estrutura que permite a comunicação de dados através de uma variável partilhada (*Shared Variable*). O uso de uma variável partilhada

permite uma abstracção na comunicação entre VIs pois é um método normalizado já usado nas anteriores versões. Para além de poderem ser usados vários tipos de dados na variável partilhada, também é possível trocar informação através de rede, entre dois computadores diferentes que possuam o LabVIEW instalado. É um método de comunicação bastante usado nos sistemas de tempo real pois é bastante escalável, suporta múltiplos clientes a aceder à mesma variável, abstrai o programador de criar código e facilita a exibição de resultados. Contudo, também possui algumas desvantagens, como, por exemplo, não ser optimizado para *stream* de dados, provocando um maior fluxo de dados em relação aos *sockets* TCP/UDP. A maior desvantagem é o facto de usar um protocolo proprietário de transferência de dados, dificultando dessa maneira a comunicação com outras plataformas e linguagens de programação.

Tal como já foi referido, o LabVIEW 2009 possui capacidade de construção de *Web Services*, oferecendo desta forma uma maior versatilidade no desenvolvimento de aplicações externas que necessitem de comunicar com programas elaborados em LabVIEW. Em vez do tradicional SOAP, o LabVIEW disponibiliza *Web Services RESTful*, que se assemelham mais a simples pedidos http. A razão desta opção prende-se com o facto de o LabVIEW trabalhar frequentemente com sistemas de tempo real, necessitando de poupar tempo na transferência de dados. Com o SOAP isso não acontece, pois existe uma definição de contrato muito mais rígida, normalmente com um XML mais longo, tornando a comunicação mais lenta. O uso de *Web Services* traz inúmeras vantagens para o funcionamento de uma aplicação:

- Usam o protocolo http, funcionando da mesma forma que um simples pedido http;
- Evitam problemas com *firewalls*, pois usam a porta 80, que normalmente está aberta para deixar passar tráfego http;
- São simples de usar, definindo um endereço com os vários parâmetros de entrada e obtendo os resultados num XML;
- O controlo de concorrência é feito automaticamente pelo servidor Web que aloja o *Web Service*;
- Suportam interacção com outras tecnologias e linguagens de programação.

Para efeitos comparativos, foi elaborada a Tabela 2 onde se vê o cruzamento entre as várias formas de comunicação referidas anteriormente e as suas características principais. Foi marcada a forma de comunicação que é superior às restantes numa determinada

tecnologia. Nos casos em que há formas de comunicação que não se destacam das restantes existem várias opções marcadas.

Tabela 2 - Cruzamento entre as formas de comunicação e suas características

	Sockets TCP/UDP	Variáveis Partilhadas	*Web Services*
Tempo de desenvolvimento		+	
Rapidez	+		
Baixa Latência	+		
Escalabilidade	+		+
Interoperabilidade			+
Uso de protocolos *standard*			+
Invocação de métodos			+
Flexibilidade			+
Eficácia	+	+	+
Complexidade		+	

Como se pode observar na Tabela 2, os *Web Services* possuem o maior número de vantagens, destacando-se dos outros dois. Alguns dos pontos menos fortes dos *Web Services* acabam por ter um baixo impacto em relação aos *sockets* e às variáveis partilhadas, como por exemplo no tempo de desenvolvimento ou na rapidez. No caso da rapidez, a desvantagem é bastante atenuada pois a comunicação com a camada intermédia é feita dentro do mesmo computador.

Desta forma, e visto que as vantagens de usar *Web Services* ofuscam claramente as desvantagens, foi decidido alterar a aplicação em LabVIEW existente de modo a que esta possa comunicar com a camada intermédia através de *Web Services* [National Instruments, 2010j].

3.3.4. Middleware

Figura 20 - Arquitectura simplificada do novo laboratório remoto: *Middleware*

Definidas as tecnologias que vão permitir à camada intermédia comunicar com o LabVIEW, é necessário escolher qual a plataforma que melhor se adequa às necessidades, as quais passam por permitir implementar a lógica de negócio e a respectiva comunicação entre a interface com o utilizador e a obtenção/envio de dados para a estação NI-ELVIS.

Numa breve investigação, chegou-se à conclusão que existem bastantes plataformas capazes de serem usadas para a implementação da camada de negócio, entre elas destacam-se PHP, Java, C#, Python, Ruby, entre outras. No entanto, são três as que

predominam: PHP, Java e C#. Normalmente, todas estas tecnologias são suportadas por servidores Web que permitem a comunicação com o exterior, como o Apache, o IIS ou Tomcat, assim como com sistemas de gestão de bases de dados, em que se destacam o SQL Server, MySQL e o Oracle.

No que diz respeito ao uso do C# para a implementação da camada intermédia, é necessário ter em conta que o servidor necessita de ter a *framework* do .NET instalada, que dependendo da versão pode ocupar entre cerca de 20Mb a 50Mb. Para além disso, seria necessário activar o servidor Web da Microsoft, o IIS, que iria suportar a comunicação com as restantes camadas. A linguagem C# já se encontra bastante madura, sendo uma das linguagens de programação mais utilizadas no mundo. Possui recursos para construir todo o tipo de aplicações, desde as tradicionais para *Desktop*, para Web e para dispositivos móveis, possuindo mecanismos de comunicação através de *Web Services*, *sockets*, *remoting* ou pedidos http. No entanto, o C# é uma linguagem proprietária da Microsoft, que exige licenças para o servidor IIS e, no caso de ser necessário o acesso a base de dados, a compra de mais uma licença de SQL Server, pois este é o sistema de gestão de bases de dados mais utilizado conjuntamente com o C#.

A plataforma Java é uma das mais conhecidas e utilizadas no mundo das tecnologias de informação e a sua linguagem de programação é a mais utilizada pelos programadores em todo o mundo. Para implementar a lógica de negócio em Java, é necessário que o servidor tenha instalado o JDK, isto é, a *framework* que suporta o desenvolvimento nessa linguagem, a qual ocupa cerca de 60Mb. A linguagem Java, tal como descrito para o C#, está presente em grande parte dos nichos de mercado, como o de dispositivos móveis, *Desktop*, Web, entre outros. Possui mecanismos de disponibilização de *Web Services*, comunicação por *sockets*, pedidos http e *remoting*. Também corre num servidor específico, que neste caso é o Apache Tomcat, que tem a vantagem de poder ser obtido gratuitamente. No que diz respeito a bases de dados, normalmente as aplicações em Java costumam ligar-se a sistemas de gestão de bases de dados como o MySQL ou Oracle, no caso de aplicações com grande fluxo de informação. Usando a plataforma Java é possível realizar toda a implementação sobre tecnologias *open source*, desde a linguagem, servidor Web e sistema de gestão de base de dados (se for escolhido o MySQL ou outro do género). No entanto, exige bastantes mais recursos do servidor, sendo a máquina virtual do Java, a JVM, conhecida por esse facto.

Em relação ao uso da tecnologia PHP, à semelhança de Java e C#, também é das linguagens mais usadas em todo o mundo, e insere-se num conjunto de tecnologias denominadas de XAMP, que normalmente inclui PHP, servidor Apache e base de dados MySQL, tanto em ambientes Windows como em Linux. A *framework* do PHP ocupa cerca de

10Mb e, tal como foi referido, usa o servidor Apache para disponibilizar as aplicações, que normalmente costumam correr apenas sobre ambientes de servidor. O PHP possui uma linguagem familiar e não é uma linguagem de programação rígida quanto ao modo de programar, ao contrário de C# e Java que são mais robustos na hora da compilação e na detecção de erros. A sua simplicidade aliada à sua leveza e ao facto de utilizar tecnologias maioritariamente *open source* permite que seja amplamente utilizada tanto por iniciantes como por profissionais. Para além disso, o facto de ser uma linguagem de *scripting* e não compilada (como Java e C#) permite a alteração "na hora" de uma aplicação, reduzindo o tempo de alteração e implementação [TIOBE, 2010].

Para efeitos comparativos, foi elaborada a Tabela 3 onde se listam as características principais das várias plataformas.

Tabela 3 - Cruzamento das plataformas com as características para o *Middleware*

	C#	**Java**	**PHP**
Linguagem/Plataforma	Proprietária	*Open Source*	*Open Source*
Multiplataforma	Não	Sim	Sim
Robustez	Grande	Grande	Boa
Rapidez	Boa	Boa	Suficiente
Facilidade de alteração	Média	Média	Fácil
Sockets	TCP e UDP	TCP e UDP	TCP e UDP
Web Services	SOAP e REST	SOAP e REST	SOAP e REST
Remoting	FluorineFx, WebOrb, AMF.NET	BlazeDS, Red5, GraniteDS, WebOrb	Zend, AMFPHP, WebOrb, SabreAMF

Decidiu-se optar por uma solução WAMP, isto é, instalação de um servidor Web Apache, PHP e MySQL no servidor de experiências que possui instalado o sistema operativo Microsoft Windows. Nesta decisão pesou, maioritariamente, o facto da aplicação em PHP poder ser alterada "na hora", sem ter de se compilar novamente o código, e o facto de não consumir tantos recursos do servidor de experiências [Adobe, 2010f].

3.3.5. Ligação entre *Middleware* e Interface

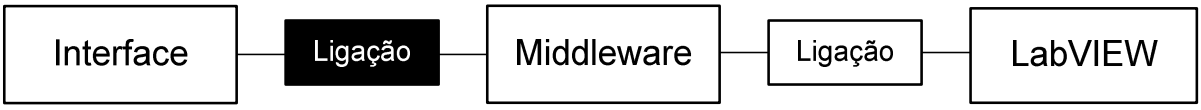

Figura 21 - Arquitectura simplificada do novo laboratório remoto: *Middleware* ↔ Interface

Uma vez escolhida a plataforma que vai implementar a lógica de negócio, é necessário definir como vai ser efectuada a ligação à interface com o utilizador. Esta ligação é bastante diferente da que foi descrita no capítulo 3.3.3 pois, em vez de comunicar com

uma camada dentro do servidor, vai comunicar com a camada de interface com o utilizador, que normalmente está noutro computador e é feita por Internet.

O PHP possui vários meios de comunicação de dados, entre eles os *sockets*, *Web Services*, *remoting* e pedidos http. O grande desafio foi encontrar a tecnologia que melhor respondesse aos requisitos de um sistema com este tipo de arquitectura, ou seja, que permitisse uma comunicação rápida, com o menor processamento possível, e que garantisse a integridade dos dados [Adobe, 2010h] [Adobe, 2010i].

Os *sockets* são suportados pela linguagem PHP desde as primeiras versões, sendo também o meio mais arcaico de comunicação de dados. Uma aplicação que comunique por *sockets* tem que estar à escuta em uma ou várias portas definidas, provocando problemas com as *firewalls*, pois podem ser usadas portas que estejam bloqueadas. Para além disso, torna a implementação mais custosa pois é necessário formatar os dados enviados e recebidos, tanto do lado do servidor como do cliente.

Os *Web Services* também estão disponíveis na linguagem PHP. As características dos *Web Services* nesta aplicação já foram referidas no capítulo 3.3.3. No entanto, para este caso, é necessário ter em conta que existe uma comunicação entre dois computadores diferentes, circulando os dados pela Internet. Essa característica faz com que o uso dos *Web Services* provoque um maior fluxo de dados, ficando a comunicação mais lenta.

O *remoting* é uma tecnologia que permite chamadas remotas por parte de um cliente a uma aplicação que se encontra no servidor. Do lado do servidor existe, normalmente, uma *framework* que funciona como *proxy*, e que permite abstrair a comunicação, serializando a informação a enviar ou a receber. Basicamente o *remoting* tem uma estrutura semelhante aos *Web Services*, possuindo métodos que são invocados pelos clientes. No entanto, o seu funcionamento é bastante diferente já que não há definição do contrato (no caso do SOAP) e é trocada a informação em formato binário, permitindo obter uma maior rapidez em relação aos *Web Services*. As aplicações que fazem uso da tecnologia Flash usam um protocolo, denominado de *Action Message Format* (AMF), que define a serialização e desserialização de objectos de dados. O AMF é considerado uma das melhores ferramentas de comunicação de dados pois, para além de ser suportado pela Adobe, é conhecido por permitir uma transferência rápida de dados, serialização de objectos entre cliente e servidor, e suportar a interoperabilidade com várias linguagens e plataformas de programação, entre elas .NET, PHP, Java, Ruby e Python. No entanto, devido ao facto de ser um protocolo proprietário, não é compatível com outras tecnologias do lado do cliente que não as da Adobe. A especificação do protocolo AMF pode ser consultada em [Adobe, 2010g].

Existe uma aplicação disponível para consulta em [James Ward, 2010] que permite testar a rapidez e o volume de dados que cada meio de comunicação possui. A Figura 22 representa o teste efectuado, dividido por tempo de execução, tempo de transferência, tempo que os dados demoram a ser tratados e tempo de renderização. Como se pode observar, e focando-se apenas na tecnologia Flex, o AMF3 é o que possui melhor desempenho. Por seu lado, os *Web Services* demoram bastante mais tempo.

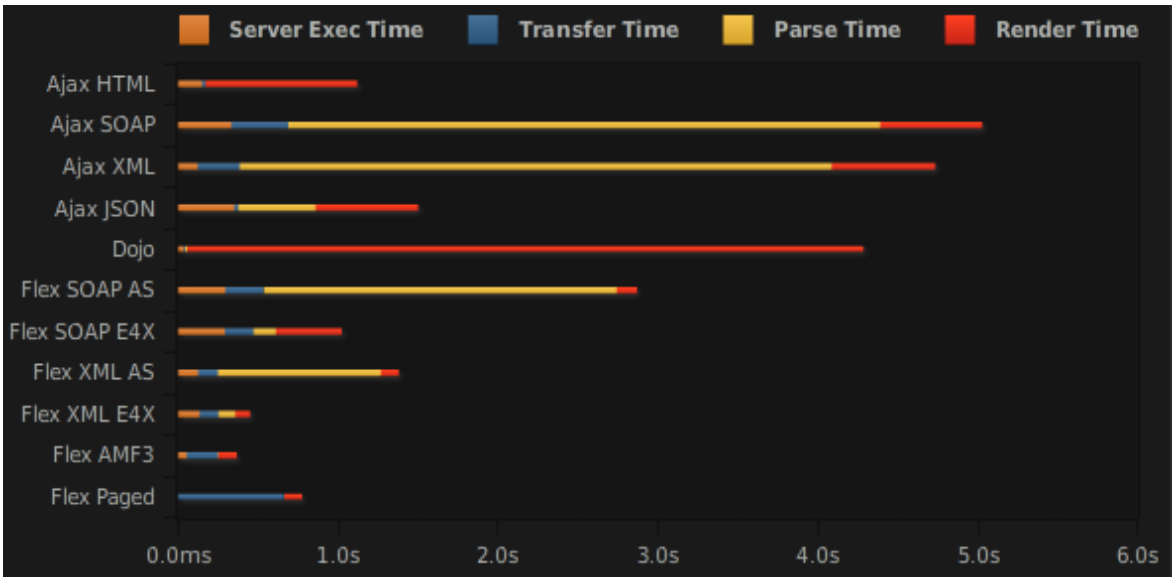

Figura 22 - Tempo gasto por cada meio de comunicação

Por seu lado a Figura 23 mostra a largura de banda consumida por cada tecnologia, onde mais uma vez o AMF3 possui as melhores prestações.

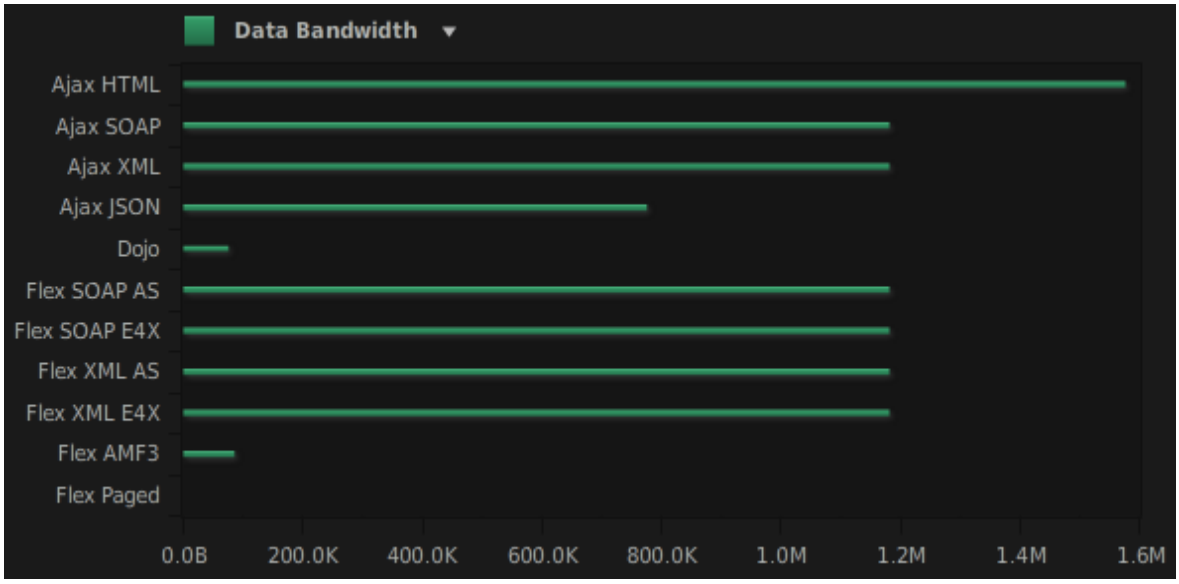

Figura 23 - Volume de dados gasto por cada meio de comunicação

Decidiu-se então optar pela solução do *remoting* devido aos consideráveis ganhos de desempenho em relação às outras tecnologias, sendo que este aspecto é bastante importante neste tipo de sistemas pois supõe-se que um laboratório remoto funcione com uma rapidez semelhante à de um laboratório real. Em relação à *framework* a usar, foi escolhida a AMFPHP pelo facto de ser bastante mais madura que as restantes (WebOrb ou Zend). Para além disso não exige instalação. Pode-se fazer *download* da *framework* e copiar a pasta para dentro do servidor Web. Possui ainda um *service browser* integrado que permite testar os serviços presentes no servidor através de uma página Web [AMFPHP, 2010].

Neste capítulo fez-se uma síntese do funcionamento do antigo laboratório remoto e o respectivo levantamento dos requisitos da nova solução, de modo a permitirem resolver os problemas do antigo laboratório remoto. Também foram justificadas todas as opções para a implementação do novo laboratório remoto. No próximo capítulo, será descrito de que modo foi implementado o novo laboratório remoto, recorrendo às tecnologias já referidas ao longo deste capítulo.

4. Implementação da nova solução

4.1. Arquitectura da solução

No anterior capítulo foram justificadas todas as acções levadas a cabo ao longo da implementação do novo laboratório remoto. Estas opções levaram à definição da arquitectura da nova solução, que está patente na Figura 24:

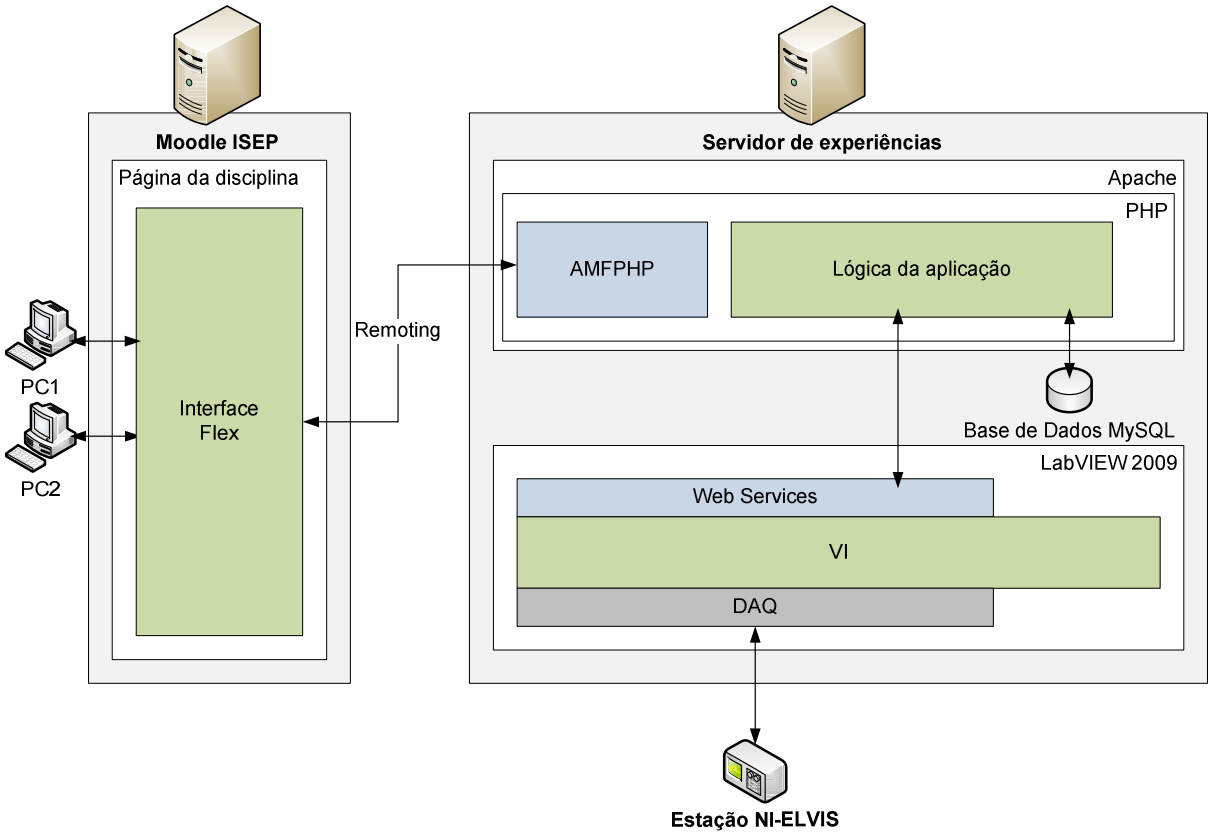

Figura 24 - Arquitectura geral do novo laboratório remoto

O utilizador vai interagir com uma camada de apresentação, que será a interface em Flex. Esta estará contida na página da disciplina presente no Moodle, de modo a facilitar o controlo de acessos (como será explicado num capítulo seguinte). A interface em Flex vai comunicar através de *remoting* com a camada intermédia, isto é, com o servidor de experiências que possui um servidor Apache. A lógica da aplicação será implementada em PHP, fazendo uso de uma base de dados para gerir os pedidos em espera. Por sua vez a camada lógica vai comunicar através de *Web Services* com o LabVIEW, que possui um VI a correr para fazer a interligação entre o LabVIEW e a estação NI-ELVIS, recorrendo ao módulo de aquisição de dados DAQ.

4.2. Implementação do laboratório

Neste subcapítulo vai ser descrito todo o processo de implementação do novo laboratório remoto. Para facilitar este processo, a descrição da implementação está dividida em vários subcapítulos.

4.2.1. Plataforma LabVIEW 2009

O primeiro passo para a implementação do novo laboratório remoto foi a instalação do LabVIEW 2009. O servidor de experiências já possuía a versão 8.5 do LabVIEW, a qual foi necessário desinstalar. Posteriormente foi efectuada a instalação do LabVIEW 2009, em que apenas se seleccionaram as funcionalidades essenciais pois, para este caso, não eram necessárias as opcionais. Após a instalação foi também necessário fazer o *download* e instalação do módulo DAQ [National Instruments, 2010k] de modo a que o LabVIEW pudesse interagir com a estação NI-ELVIS ligada ao servidor de experiências. Após a instalação, era possível utilizar o antigo laboratório, permitindo desta maneira a alteração da aplicação em LabVIEW.

Tal como já foi descrito, o laboratório antigo possuía vários VIs a correr em separado, e cada um tinha funções de comunicação com o Moodle e de gestão dos pedidos efectuados pelos clientes para comunicar com a estação NI-ELVIS. Quando um cliente fazia o pedido, era aberto um VI remoto com uma interface para ser controlada através do *Web Browser*. No entanto, sentiu-se a necessidade de melhorar este processo porque podiam haver vários painéis remotos abertos em simultâneo e sobrecarregar o servidor. Desta forma, visto que vão ser usados *Web Services* para definir os valores de configuração e obter os resultados, decidiu-se alterar o VI que continha o painel remoto de modo a este estar constantemente a correr e servir para todos os pedidos efectuados. Assim, o servidor de experiências apenas tem a carga máxima de um painel aberto, ao contrário dos vários painéis possíveis anteriormente.

A primeira modificação efectuada no VI foi mudar a zona onde se obtêm os resultados de modo a colocá-los numa estrutura em formato de *array*, para que mais tarde fosse utilizada para gerar uma saída de XML para os *Web Services*. Os valores das tensões e correntes obtidos pela estação NI-ELVIS já se encontravam numa estrutura com formato de *array*, pelo que foi feita a respectiva réplica para uma estrutura semelhante. Os valores do canal A e canal B do osciloscópio, que eram mostrados graficamente, foram igualmente convertidos para uma estrutura do mesmo tipo.

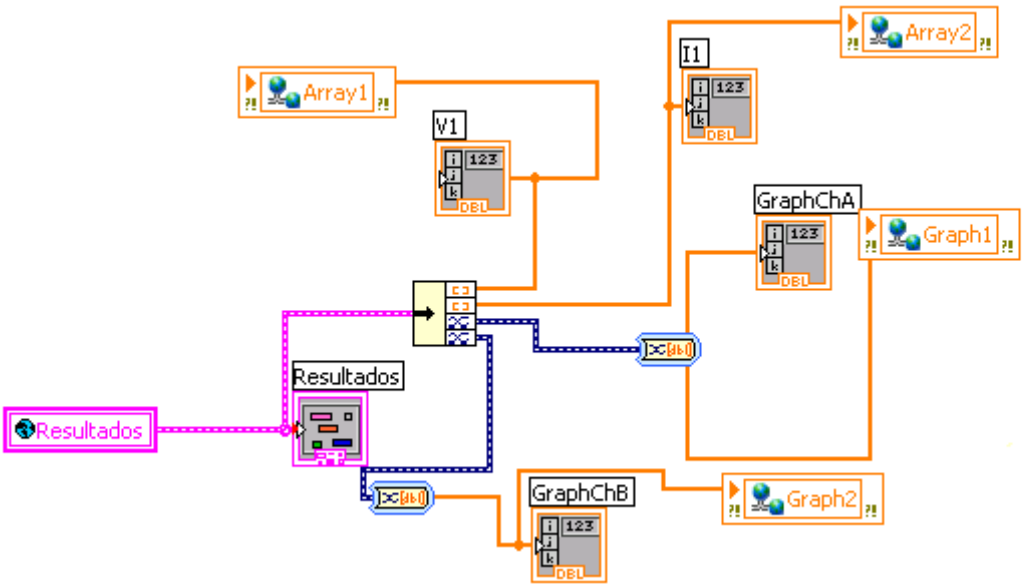

Figura 25 - *Block Diagram* alterado para os valores obtidos da estação NI-ELVIS

Como se pode observar na Figura 25, os resultados eram obtidos numa estrutura do tipo “cluster”. Para que os valores sejam separados foi necessário converter o “cluster” nas várias estruturas que o compõem, através do objecto “Unbundle” presente nas ferramentas do LabVIEW. Cada tipo de valores obtidos foi convertido numa estrutura do tipo *array* (V1, I1, GraphChA e GraphChB), ficando associada a cada um deles uma variável partilhada. As variáveis partilhadas são estruturas da linguagem LabVIEW que permitem partilhar informação por toda a aplicação, com um funcionamento idêntico às variáveis globais na maioria das linguagens de programação. O uso das variáveis partilhadas, para este caso, prende-se principalmente com o uso de *Web Services*, mas este é um assunto que será explicado posteriormente.

Após a alteração da zona de obtenção dos dados, foi necessário definir a forma como se introduzem os valores de configuração. Após uma série de testes com os componentes existentes no LabVIEW, chegou-se à conclusão que podiam ser usadas as estruturas presentes na aplicação actual, bastando para isso definir o valor de cada uma. Para esse efeito foram usadas, mais uma vez, variáveis partilhadas, cujo valor é definido através da invocação de *Web Services* e está ligado ao valor da respectiva estrutura.

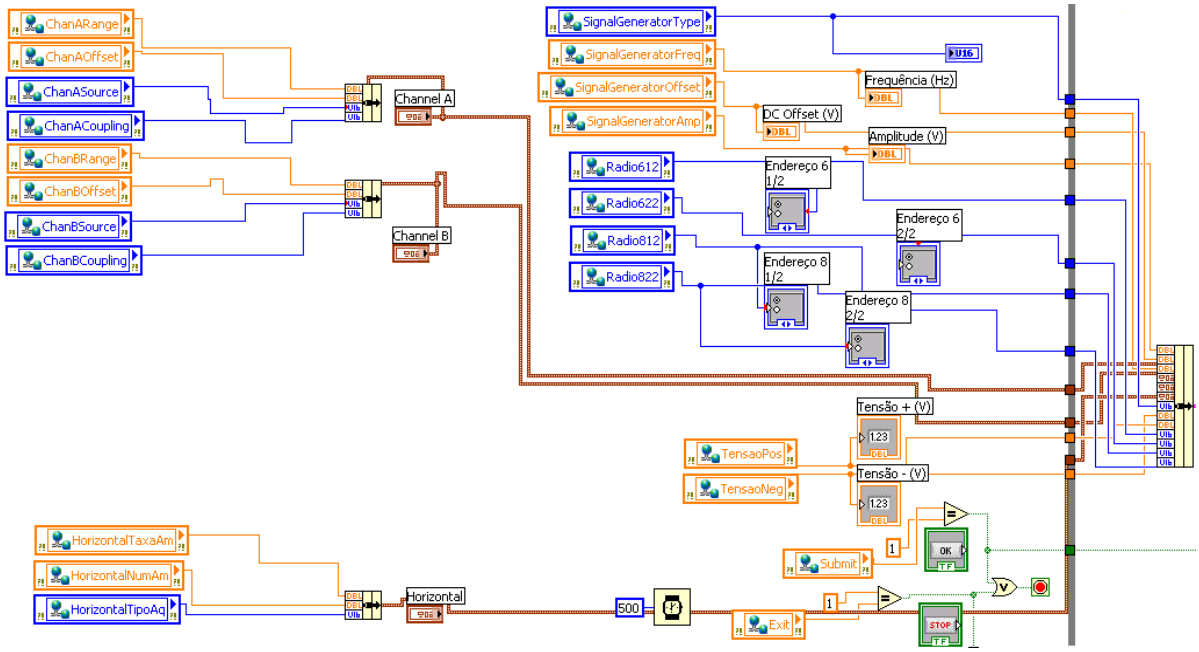

Figura 26 - *Block Diagram* alterado para os valores de configuração

Tal como o demonstra a Figura 26, algumas variáveis partilhadas foram ligadas a uma estrutura do tipo “Bundle”, de forma a juntar alguns parâmetros que estão no mesmo “cluster”. Outras variáveis ligam-se directamente com a respectiva estrutura, que por sua vez se liga ao “cluster” principal, que alberga todos os parâmetros de configuração, para posteriormente os passar à estação NI-ELVIS.

Nota ainda para a modificação feita às acções dos botões “Submeter” e “Sair” presentes na interface com o utilizador. Como o painel passou a ser controlado todo ele por *Web Services* através de variáveis partilhadas, foram também introduzidas duas variáveis para controlar a submissão de cada pedido e o respectivo cancelamento. Desta forma, na interface manteve-se o botão “Submeter” usado para submeter um novo pedido para a estação NI-ELVIS tratar. O botão “Sair” deixou de fazer sentido pois no novo laboratório não vai ser necessário fechar o VI de ligação à estação NI-ELVIS, sendo substituído pelo botão “Cancelar” que aparece numa nova janela sobreposta à anterior, e que permite cancelar o pedido efectuado anteriormente. A interface do painel frontal foi modificada de modo a esconder as estruturas de definição dos valores de configuração, pois não fazia sentido existirem se o utilizador não vai definir esses valores na interface do LabVIEW. Os valores obtidos e os gráficos foram deixados no painel frontal, para o caso do administrador do servidor de experiências sentir necessidade de elaborar testes de despistagem de erros se surgir algum problema com o laboratório. Uma vez alterado o painel principal, foi necessário efectuar o restante desenvolvimento sobre a plataforma LabVIEW. Devido à necessidade de uso de *Web Services* e variáveis partilhadas, foi criado um novo projecto LabVIEW 2009. A estrutura dos novos projectos LabVIEW obedece a uma certa ordenação, colocando-se

normalmente os VIs na raiz do projecto. As variáveis partilhadas, dependências de VI, pastas virtuais e especificações de *Web Services* ficam num local predefinido para eles.

Sendo assim, foi criado um VI na raiz do projecto denominado de "caller.vi", onde foram inseridas as instruções e fluxos do VI alterado, o qual foi descrito anteriormente. Este VI é o que vai ficar sempre a correr e fazer o pedido à estação NI-ELVIS, por isso foi-lhe dado esse nome. Após isso foram criadas as variáveis partilhadas que, tal como foi referido, estão localizadas numa estrutura especial criada para esse efeito, uma biblioteca chamada de "SharedVar.lvlib". Foi criada uma variável partilhada por cada valor de configuração, assim como quatro variáveis por cada tipo de valor resultante: tensões; correntes; e valores dos gráficos. Na Tabela 4 pode-se observar a lista completa das variáveis partilhadas que foram criadas, assim como uma breve descrição de cada uma.

Tabela 4 - Lista de variáveis partilhadas criadas

Nome da variável	Descrição
Radio612	Endereço 6: 1/2
Radio622	Endereço 6: 2/2
Radio812	Endereço 8: 1/2
Radio822	Endereço 8: 2/2
SignalGeneratorFreq	Gerador de Sinal: Frequência
SignalGeneratorOffset	Gerador de Sinal: Offset
SignalGeneratorType	Gerador de Sinal: Tipo
SignalGeneratorAmp	Gerador de Sinal: Amplitude
ChanARange	Canal A: Gama
ChanAOffset	Canal A: Offset
ChanASource	Canal A: Fonte
ChanACoupling	Canal A: Acoplamento
ChanBRange	Canal B: Gama
ChanBOffset	Canal B: Offset
ChanBSource	Canal B: Fonte
ChanBCoupling	Canal B: Acoplamento
HorizontalTaxaAm	Horizontal: Taxa de amostragem
HorizontalNumAm	Horizontal: Número de amostras
HorizontalTipoAq	Horizontal: Tipo de Aquisição
TensaoPos	Fontes de Tensão Reguláveis: Tensão +
TensaoNeg	Fontes de Tensão Reguláveis: Tensão -
Submit	Controlo de submissão
Exit	Cancelar submissão
Array1	Tensões registadas V1
Array2	Correntes registadas I1
Graph1	ELVIS Canal A
Graph2	ELVIS Canal B

Cada variável partilhada está associada a um valor de configuração, e é alterada através da chamada de um *Web Service*, que foi desenvolvido para esse efeito.

Porém, antes da construção do *Web Service*, foi necessário construir alguns VIs com as estruturas simbólicas dos valores de configuração. Podia ter sido construído apenas um VI para esse efeito. No entanto, o desenvolvimento ficava mais confuso, pelo que se optou pela construção de vários VIs, cada um contendo as variáveis da mesma área. Na Tabela 5 pode-se observar a forma como foi feita a divisão. Cada VI possui várias estruturas que simbolizam um valor de configuração, tendo cada estrutura a respectiva variável partilhada ligada a ela.

Tabela 5 - VIs criados para a construção do *Web Service*

VI	Nome da variável	Valor de configuração
shared.vi	Radio612	Endereço 6: 1/2
	Radio622	Endereço 6: 2/2
	Radio812	Endereço 8: 1/2
	Radio822	Endereço 8: 2/2
sharedsignal.vi	SignalGeneratorFreq	Gerador de Sinal: Frequência
	SignalGeneratorOffset	Gerador de Sinal: Offset
	SignalGeneratorType	Gerador de Sinal: Tipo
	SignalGeneratorAmp	Gerador de Sinal: Amplitude
sharedchana.vi	ChanARange	Canal A: Gama
	ChanAOffset	Canal A: Offset
	ChanASource	Canal A: Fonte
	ChanACoupling	Canal A: Acoplamento
sharedchanb.vi	ChanBRange	Canal B: Gama
	ChanBOffset	Canal B: Offset
	ChanBSource	Canal B: Fonte
	ChanBCoupling	Canal B: Acoplamento
sharedhorizontal.vi	HorizontalTaxaAm	Horizontal: Taxa de amostragem
	HorizontalNumAm	Horizontal: Número de amostras
	HorizontalTipoAq	Horizontal: Tipo de Aquisição
sharedtensao.vi	TensaoPos	Fontes de Tensão Reguláveis: Tensão +
	TensaoNeg	Fontes de Tensão Reguláveis: Tensão -
ws.vi	Submit	Controlo de submissão
	Exit	Cancelar submissão
data.vi	Array1	Tensões registadas V1
	Array2	Correntes registadas I1
	Graph1	ELVIS Canal A
	Graph2	ELVIS Canal B

Todas as variáveis partilhadas, excepto as que estão no VI “data.vi”, são de escrita. Ou seja, a estrutura escreve na variável o valor definido. A título exemplificativo vai ser explicado o processo de construção do VI “sharedchana.vi“, que possui as estruturas de

definição para o canal A. Após criar um novo VI dentro do projecto, são inseridas no painel frontal as estruturas que representam os quatro valores de configuração. A Figura 27 mostra a estrutura do VI. No painel frontal foi usada a estrutura "Numeric" do LabVIEW para cada valor de configuração, pois os valores são numéricos. No *block diagram* foram ligadas, a cada estrutura, uma variável partilhada de escrita, a qual vai armazenar um valor da cada vez que o *Web Service* deste VI for chamado.

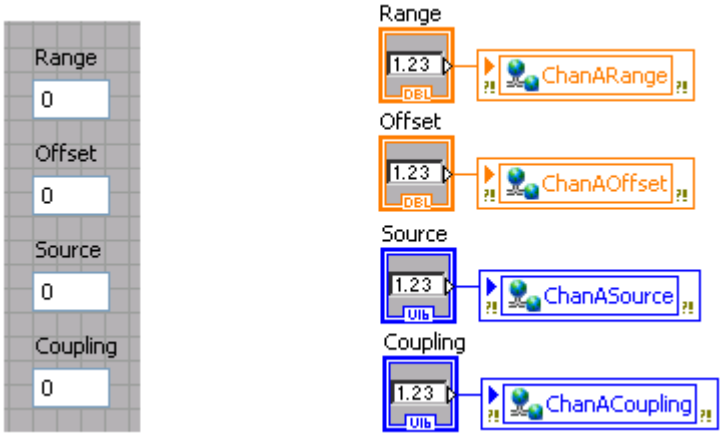

Figura 27 - *Front Panel* (esquerda) e *Block Diagram* (direita) do VI sharedchana.vi

Para além deste conteúdo inserido no VI é necessário indicar se as estruturas são para usar como variáveis de configuração ou se o seu valor fará parte do resultado do XML. Para esse efeito é necessário, na parte superior direita do painel frontal e através do botão direito do rato, escolher a opção "Show Connector" que vai mostrar uma tabela do género da Figura 28.

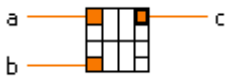

Figura 28 - *Connector Pane* na construção de um *Web Service*

O *Connector Pane* está dividido em vários quadrados. Nos quadrados da esquerda definem-se as estruturas associadas aos valores de configuração definidos através da chamada de um *Web Service*. Nos da direita definem-se as variáveis que possuem os valores a aparecer no XML de resposta quando um *Web Service* é invocado. No caso da Figura 28, as variáveis "a" e "b" são os valores de entrada e a variável "c" a que possui o valor a aparecer no XML final. São definidas seleccionando o respectivo quadrado e logo após seleccionar a estrutura correspondente. Nota ainda para o facto de ser possível adicionar espaço para mais variáveis de entrada ou de saída, até ao limite de oito.

No caso do VI "sharedchana.vi", como apenas existem variáveis de entrada o seu aspecto ficou semelhante ao que mostra a Figura 29.

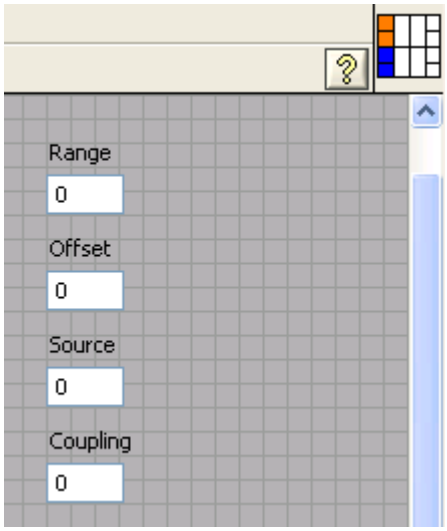

Figura 29 - *Front Panel* com o *Connector Pane* visível e as variáveis definidas

Todos os VIs, excepto o "data.vi", seguem a mesma lógica de construção descrita anteriormente. O VI "data.vi" é usado para obter os valores resultantes da medição na estação NI-ELVIS, por isso a sua implementação é diferente das dos outros VIs. As variáveis partilhadas ligadas às estruturas são de leitura, e o *Connector Pane* reflecte apenas o mapeamento das variáveis cujo valor vai ser reflectido no XML gerado pelo *Web Service*.

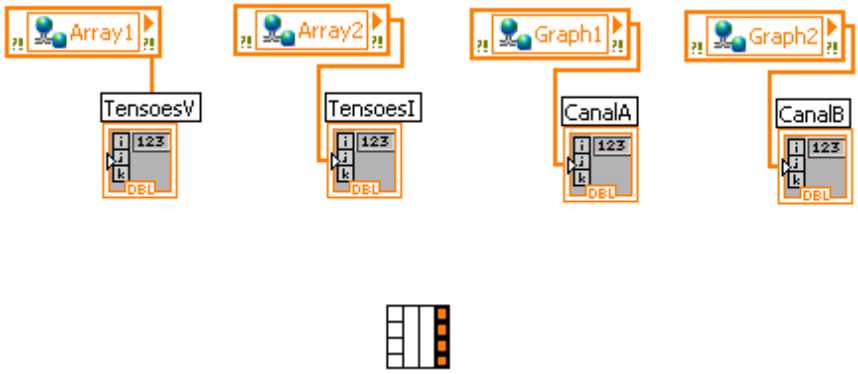

Figura 30 - *Block Diagram* (em cima) e *Connector Pane* (em baixo) do VI "data.vi"

Uma vez definidos os VIs, é necessária a construção do *Web Service* propriamente dito. Para isso existe uma pasta dentro do projecto chamada de "Build Specification", que alberga todos os *Web Services* criados no projecto. Assim, escolheu-se a opção de criar um novo "Web Service (RESTful)" [National Instruments, 2010l], que abriu uma nova janela com opções de configuração. As opções estão divididas por categorias, em que as mais importantes são:

- "**Information**" – Nesta categoria escolhe-se o nome a atribuir ao *Web Service* do projecto, o nome do serviço e a localização dos ficheiros no computador

local. Foi dada a designação “devservice” ao nome do serviço. Os restantes campos são irrelevantes;

- “**Source Files**” – Nesta categoria aparecem listados todos os VIs que existem no projecto. É necessário escolher todos os VIs que possuam dados que estejam relacionados com o *Web Service*, passando-os para uma área chamada de “Service VIs”. Neste caso foram seleccionados todos os VIs elaborados de propósito para a interacção de dados com o *Web Service*, ou seja: “ws.vi”, “data.vi”, “shared.vi”, “sharedchana.vi”, “sharedchanb.vi”, “sharedhorizontal.vi”, “sharedsignal.vi” e “sharedtensao.vi”. De referir ainda que ao escolher cada VI pode-se definir se o mesmo é mantido sempre em memória, se é corrido sempre quando o servidor inicia e o tipo de resposta pretendida (XML, texto, HTML ou JSON);
- “**URL Mappings**” – Nesta categoria define-se o mapeamento dos endereços para interagir com cada VI. Normalmente o endereço possui o nome do VI seguido das variáveis definidas no *Connector Pane*. Também é possível definir que método http é usado quando o serviço é chamado, como por exemplo GET ou POST [National Instruments, 2010m].

As restantes categorias servem para definições comportamentais do *Web Service*, como, por exemplo, definição de palavra passe, compatibilidade dos VIs com antigas versões, local de geração de *logs*, entre outros. Uma vez definidas todas as opções, é necessário confirmar e clicar em “*Build*” para que o projecto compile. Caso não existam erros pode-se clicar em “*Deploy*”, isto é, disponibilizar no servidor Web do LabVIEW o *Web Service* criado. Resta referir que o servidor Web do LabVIEW deve estar activo, e para isso pode-se ir ao menu “Tools”, “Options” e verificar se a opção “Enable Web Server” do separador “Web Server” está marcada [National Instruments, 2010n].

Uma vez construído o *Web Service*, este pode ser invocado com um simples pedido http, bastando para isso invocar o nome do serviço seguido dos seus parâmetros. Tal como foi explicado anteriormente, o serviço possui normalmente o nome do VI a seguir ao nome do *Web Service*, podendo opcionalmente indicar os parâmetros de entrada. Normalmente um pedido tem a seguinte estrutura:

```
http://<Endereço do servidor Web>/<Nome do Web Service>/<Nome do VI>/<Parâmetros>
```

No caso do projecto em causa, o endereço do servidor Web é **localhost** pois o servidor corre localmente no IP 127.0.0.1 na porta 80. O nome do *Web Service* já foi referido anteriormente: **devservice**. Na Tabela 6 estão contidos todos os endereços mapeados para

interagir com os VIs do *Web Service*. O pedido é efectuado juntando o endereço da coluna da esquerda com um da coluna "URL Mappings". As palavras que começam por dois pontos simbolizam os parâmetros de entrada, que podem ou não ser definidos na requisição.

Tabela 6 - Lista dos URL *Mappings* definidos para o *Web Service*

URL	URL *Mappings*
http://localhost/devservice	/ws/:Submit
	/shared/:Radio612/:Radio622/:Radio812/:Radio822
	/data
	/sharedsignal/:Type/:Freq/:Offset/:Amp
	/sharedchana/:Range/:Offset/:Source/:Coupling
	/sharedchanb/:Range/:Offset/:Source/:Coupling
	/sharedhorizontal/:TaxaAm/:NumAm/:TipoAq
	/sharedtensao/:Pos/:Neg

4.2.2. Lógica de negócio

Uma vez preparada toda a plataforma LabVIEW, foi necessário implementar a camada intermédia, ou seja, a lógica de negócio. Tal como havia sido justificado no capítulo 3.3.4, optou-se por implementar a lógica de negócio sobre uma plataforma WAMP, isto é, usando a linguagem de programação PHP no servidor Apache e recorrendo ao sistema de gestão de bases de dados MySQL. Estas tecnologias podem ser instaladas separadamente, ou então, e visto que são *open source*, obter um pacote com todas essas tecnologias no site do *Apache Friends* [Apache Friends, 2010], onde se pode fazer *download* de um arquivo com a última versão estável dessas tecnologias. Assim, foi feito o *download* e descompactado o arquivo para uma pasta dentro do servidor de experiências. Cada tecnologia pode ser configurada à medida, sem depender da outra, por isso não houve qualquer reticência ao instalar este pacote ao invés de instalar cada plataforma em separado. Os tópicos seguintes apresentam uma breve descrição das configurações efectuadas em cada uma das plataformas:

- **Apache** – O Apache é o servidor Web, a par do IIS da Microsoft, mais conhecido em todo o mundo. É bastante configurável e a sua versão estável, e instalada no servidor de experiências, é a 2.2.

  Por omissão o servidor está configurado para correr na porta 80, mas visto que o servidor Web do LabVIEW também corre nessa porta houve necessidade de alterar a porta, que passou a ser a 8080. Assim, editou-se o ficheiro "httpd.conf" e alterou-se essa definição:

  ```
  Listen 8080
  ```

Para além disso resta destacar que é neste ficheiro que se define que tipo de acesso tem uma pasta ou um ficheiro a partir do exterior. Normalmente os sites ficam dentro da pasta "htdocs" mas a sua localização pode ser alterada na linha:

```
DocumentRoot "C:/xampp/htdocs"
```

- **PHP** – A versão do PHP instalada foi a 5.3, que já possui capacidades de programação por objectos. À semelhança do Apache, também o PHP possui um ficheiro de configuração, que neste caso se chama "php.ini".

  Foram feitas algumas alterações no ficheiro "php.ini". No início da pesquisa, para fazer os testes com *sockets*, foi necessário definir o suporte a estes:

  ```
  extension=php_sockets.dll
  ```

  Também foi necessário definir o tempo máximo de execução de um *script* em PHP. Normalmente está definido para 60 segundos, mas como será explicado posteriormente houve necessidade de aumentar o valor para 3600 segundos:

  ```
  max_input_time = 3600
  ```

- **MySQL** – O MySQL é um sistema de gestão de base de dados leve mas bastante poderoso para sistemas deste tipo. A versão instalada foi a 5.1. Possui o ficheiro de configuração "my.ini", mas como corre na porta 3306 não houve necessidade de qualquer alteração, excepto da palavra passe que por omissão vem nula e foi alterada por razões de segurança.

Para além do que foi descrito anteriormente, com o WAMP vem uma plataforma que permite gerir as bases de dados MySQL através de uma interface Web, o PHPMyAdmin. Esta plataforma foi bastante útil ao longo do projecto pois facilitou a interacção com a base de dados nos vários testes efectuados.

Uma vez que o WAMP deve estar sempre a correr no servidor de experiências, foi definido que o programa iria iniciar automaticamente no arranque do sistema operativo. Para além disso é possível iniciar ou parar os programas do Apache e do MySQL. Também existe possibilidade dos dois programas correrem como processos do sistema operativo, abrindo automaticamente no arranque do servidor. Como o projecto exigia esse requisito, foi definido que o Apache e o MySQL iriam correr como processos no arranque do sistema operativo.

De modo a armazenar os dados e tratar os pedidos efectuados, foi criada uma base de dados recorrendo à ferramenta PHPMyAdmin, com o nome "remotelab". Esta base de dados tem a função de registar todos os pedidos efectuados por clientes, assim como dar a

conhecer quais os pedidos que foram já tratados pela estação NI-ELVIS. Para além disso, permite implementar com maior segurança uma lista de espera virtual, de forma a tratar os pedidos efectuados pelos clientes pela ordem de chegada. De forma a satisfazer esses requisitos, foi identificada a necessidade de criar uma tabela com alguns campos, os quais são mostrados na Tabela 7.

Tabela 7 - Estrutura da tabela "requisition" da base de dados "remotelab"

Campo	Tipo	Descrição
idRequisition	Inteiro	Identificador único de um pedido
radio1	Texto	Valor de configuração
radio2	Texto	Valor de configuração
radio3	Texto	Valor de configuração
radio4	Texto	Valor de configuração
signalSignal	Texto	Valor de configuração
signalFrequency	Texto	Valor de configuração
signalOffset	Texto	Valor de configuração
signalAmplitude	Texto	Valor de configuração
channelaSource	Texto	Valor de configuração
channelaRange	Texto	Valor de configuração
channelaCoupling	Texto	Valor de configuração
channelaOffset	Texto	Valor de configuração
channelbSource	Texto	Valor de configuração
channelbRange	Texto	Valor de configuração
channelbCoupling	Texto	Valor de configuração
channelbOffset	Texto	Valor de configuração
horizontalRate	Texto	Valor de configuração
horizontalSamples	Texto	Valor de configuração
horizontalRateAcq	Texto	Valor de configuração
tensaoPos	Texto	Valor de configuração
tensaoNeg	Texto	Valor de configuração
requisitionDate	Data	Data em que foi efectuada a requisição
accessDate	Data	Última data de alteração ao pedido
resultData	Texto longo	XML com os resultados

A maioria dos campos tem a função de armazenar os valores das variáveis de configuração, com excepção de quatro deles: o "idRequisition", que funciona como chave primária da tabela e permite numerar os pedidos à medida que são inseridos na base de dados; o campo "resultData", que permite armazenar o XML proveniente da chamada do *Web Service* em LabVIEW, o qual devolve os resultados obtidos da estação NI-ELVIS; e, por fim, dois campos de armazenamento de datas: "accessDate", que guarda a data do último acesso ao registo; e "requisitionDate", que armazena a data de inserção do registo, e permite também tratar os pedidos segundo a ordem de chegada.

Uma vez configurado o servidor Web e a base de dados, partiu-se para a implementação utilizando a linguagem PHP, começando-se por fazer o *download* do AMFPHP e por colocar a directoria dentro da pasta "htdocs" do Apache, de forma a poder ser acedido. O AMFPHP é composto por várias partes: possui a implementação da plataforma; uma zona para aceder ao *service browser*, de forma a testar os serviços desenvolvidos; e outro local para colocar os ficheiros PHP com os respectivos serviços, dentro da pasta "services". Assim, criou-se um ficheiro chamado "RemoteLab.php" que possui uma classe com o mesmo nome, dentro da qual está a definição dos métodos remotos. Também foi criado um ficheiro "config.php" que contém as configurações relativas à ligação com a base de dados, como o nome da base de dados, utilizador e palavra passe. É normal, nestes casos, existir um ficheiro separado com estas configurações, pois desta forma este é incluído em todos os ficheiros da aplicação que necessitem de aceder à base de dados, evitando definir várias vezes as mesmas configurações.

Os métodos AMFPHP, a implementar na classe "RemoteLab", funcionam de forma idêntica ao dos *Web Services*. Isto é, o método é chamado pelo cliente e este recebe uma resposta logo que possível. No entanto, para esta implementação, quando um cliente faz um pedido, este fica numa lista de espera, o que faria com que o cliente ficasse à espera da resposta durante um período de tempo indeterminado. Para resolver essa situação decidiu-se criar três métodos, um para submeter o pedido na lista de espera, outro para verificar se o pedido já foi tratado e outro para cancelar o pedido. Para além disso houve necessidade de criar um *script* em paralelo que vai executar todos os pedidos que estão em lista de espera.

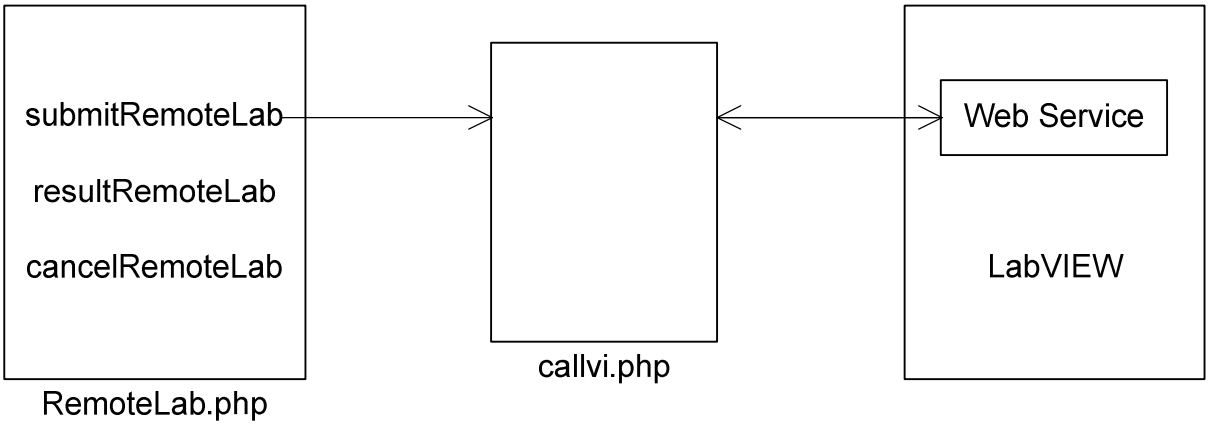

Figura 31 - Estrutura de ficheiros em PHP

Para um cliente submeter um novo pedido, tem de chamar o método "submitRemoteLab" da classe "RemoteLab". Este método recebe como argumento um objecto com todos os valores de configuração e começa por interrogar a base de dados

sobre qual foi o último registo inserido, de forma a obter o identificador do novo pedido. A seguir, o pedido é colocado na lista de espera, ou seja, é inserido um novo registo na base de dados com os respectivos valores de configuração. Após isso, é feita uma consulta à base de dados para saber quantos pedidos estão em lista de espera, de modo a fazer uma estimativa de quanto tempo vai demorar o novo pedido a ser tratado pela estação NI-ELVIS. Por fim, é executado o ficheiro "callvi.php", que será explicado a seguir. O método devolve ao cliente um objecto com o identificador do pedido (para mais tarde requerer os resultados) e com uma estimativa do tempo de espera. O *workflow* descrito pode ser observado na Figura 32.

O ficheiro "callvi.php" é chamado cada vez que é inserido um novo pedido, sendo esta a melhor solução encontrada de modo a não deixar o cliente com uma ligação pendente quando não se sabia quanto tempo iria esperar. O *script* começa por verificar, através de uma variável, se este já está a ser executado por outro cliente, e, caso afirmativo, interrompe a sua execução, de forma a garantir que não existem múltiplas instâncias a correr ao mesmo tempo. Também foi previsto o caso de o servidor de experiências ou outro programa qualquer falhar, tendo deixado a variável no estado de ocupada. Neste caso seria impossível voltar a executar o *script*. Por isso, decidiu-se implementar outra variável que armazena a data de início de execução, permitindo a execução do ficheiro se o último pedido ocorreu há mais de 5 minutos. Caso seja possível avançar, a primeira instrução a executar é a de bloqueio da execução do *script* a outros pedidos. De seguida faz uma consulta à base de dados para saber se existem pedidos na lista de espera. Caso existam, faz outra consulta para obter os dados do pedido que está na cabeça da fila de espera. Os valores de configuração são usados para a definição dos pedidos ao *Web Service* do LabVIEW. Após definidos os valores de configuração, é dada ordem para que a estação NI-ELVIS trate esse pedido. Após o período de espera, são pedidos os resultados através de uma chamada ao *Web Service*, resultados esses que são inseridos num campo do respectivo registo na base de dados. Por fim, é verificado se existem mais pedidos em lista de espera. Em caso afirmativo, repete-se o processo descrito anteriormente até não existirem pedidos pendentes. Quando já não houver pedidos em lista de espera, o bloqueio de execução ao ficheiro "callvi.php" é libertado, terminando dessa forma o seu funcionamento. O *workflow* descrito pode ser observado na Figura 33.

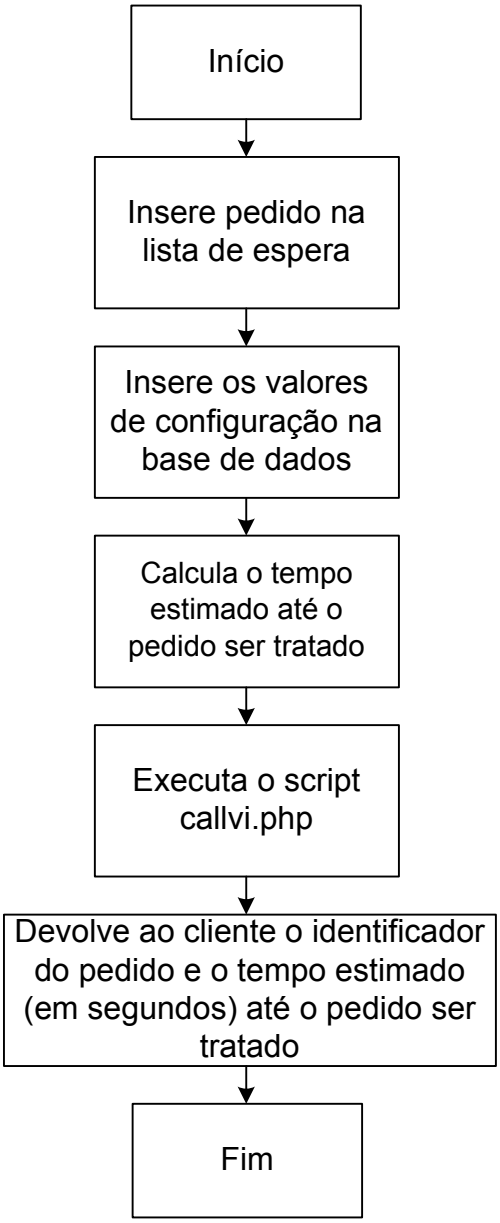

Figura 32 - *Workflow* do método "submitRemoteLab"

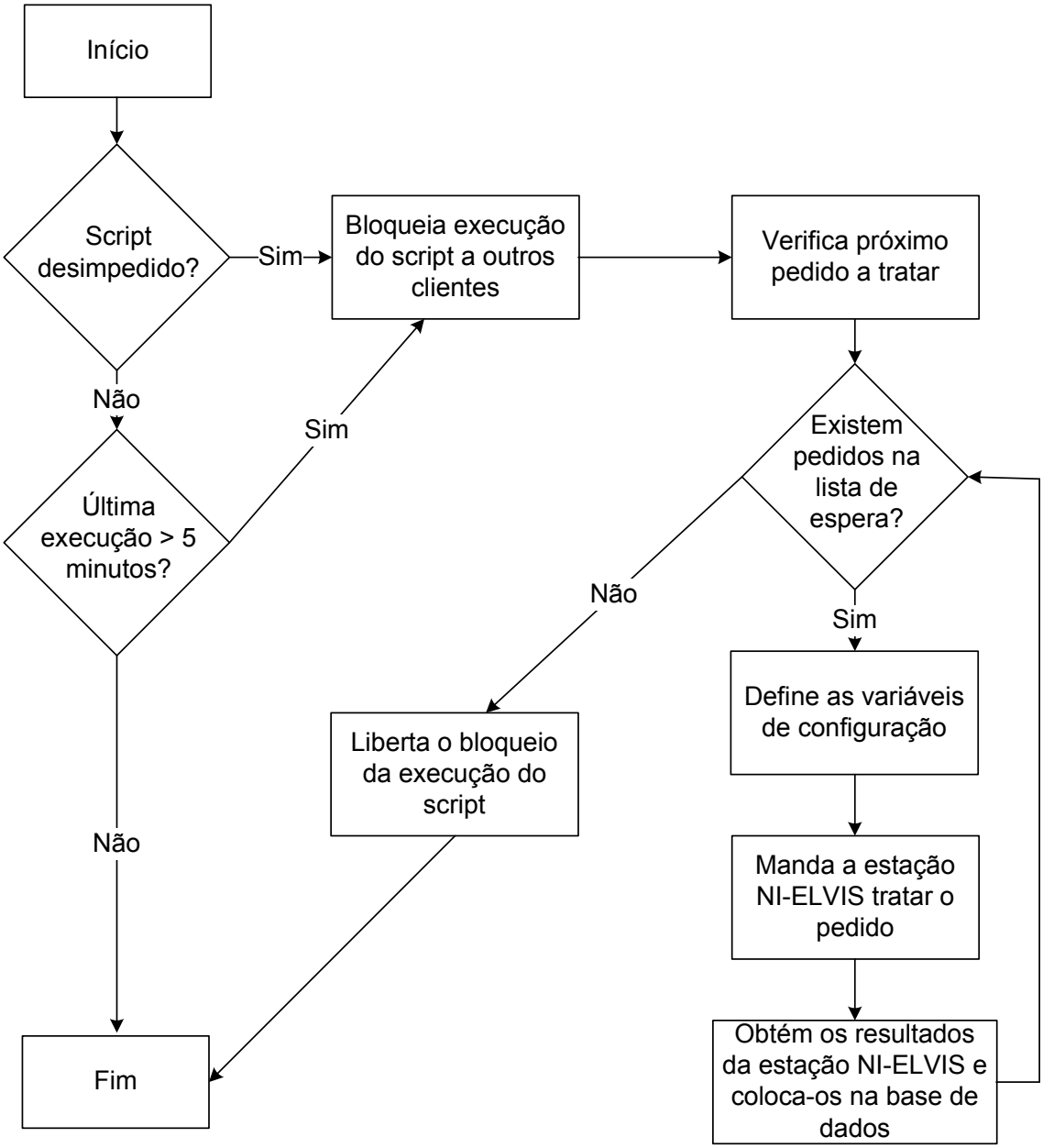

Figura 33 - *Workflow* do ficheiro "callvi.php"

O outro método da classe “RemoteLab” é o que vai permitir aceder aos resultados de um pedido já tratado pela estação NI-ELVIS. O “resultRemoteLab” recebe como argumento o identificador do pedido e verifica de seguida com uma consulta à base de dados se os resultados já estão disponíveis. Se já estiverem disponíveis, é enviado para o cliente um objecto com o XML dos dados medidos pela estação NI-ELVIS. O *workflow* descrito pode ser observado na Figura 34.

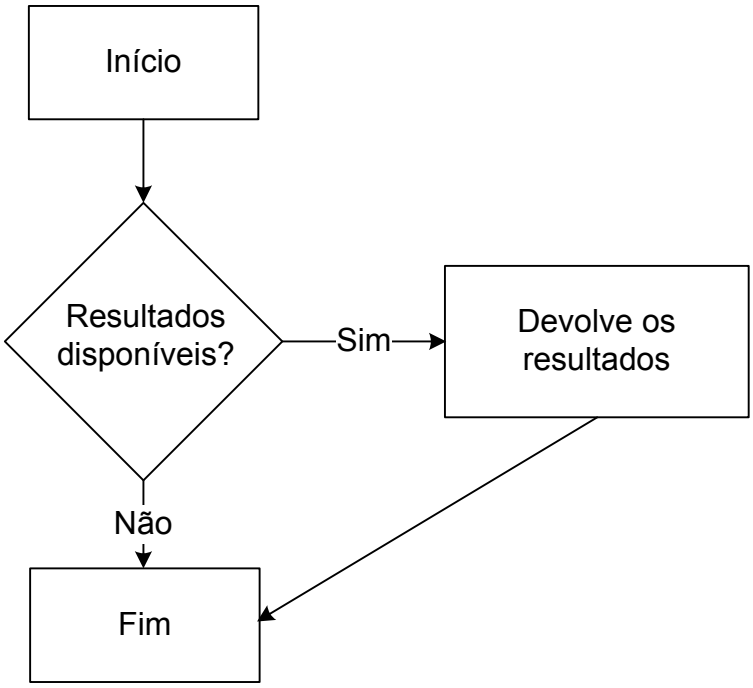

Figura 34 - *Workflow* do método "resultRemoteLab"

Se os resultados ainda não estiverem disponíveis, o pedido é repetido posteriormente a intervalos de 10 em 10 segundos, até que os resultados sejam recebidos com sucesso.

Por fim, existe ainda outro método chamado “cancelRemoteLab”, cuja função é retirar da lista de espera um pedido cancelado pelo utilizador. Para isso, o método recebe como argumento o identificador do pedido, marcando na base de dados que o pedido com esse identificador foi cancelado.

4.2.3. Interface Flex

A construção do interface da aplicação em Flex é o desenvolvimento cuja opção teve mais peso na decisão de elaborar um novo laboratório remoto. A nova interface tinha como objectivo ser capaz de resolver os principais problemas detectados no antigo laboratório, como, por exemplo, o funcionamento em várias plataformas, e assegurar uma maior interoperabilidade. A interface mereceu preocupações com a usabilidade e com a sequência das operações, de modo a facilitar a interacção com o utilizador. A interface em Flex foi desenvolvida com recurso ao Flex 3 SDK, uma biblioteca em Action Script 3 com grandes capacidades de produtividade e desempenho. O primeiro passo foi a instalação do IDE Flex Builder 3, sendo feito o *download* do IDE Eclipse e do *plug-in* disponibilizado no site da Adobe. A pasta do Eclipse apenas requer ser copiada para uma localização específica, que será indicada durante a instalação do *plug-in* do Flex Builder 3. A instalação também inclui o Flex 3 SDK, evitando dessa forma ter de se fazer *download* da biblioteca. Após a instalação estar concluída, partiu-se para a esquematização do projecto.

O primeiro passo na construção do programa é sempre criar o projecto. Nas opções do menu do IDE clicou-se na opção de criar um novo “Flex Project”, aparecendo de seguida uma janela com algumas opções, entre elas o nome do projecto. Neste caso foi atribuído o nome “RemoteLab”, deixando as outras opções com os valores por omissão. Após clicar no botão “Concluir”, o novo projecto está pronto a ser desenvolvido, sendo criado automaticamente um ficheiro MXML. Convém relembrar que os ficheiros MXML servem para construir a interface da aplicação, através da inserção de *tags* XML que representam os componentes da biblioteca. Contudo, também é possível inserir código Action Script 3 dentro do ficheiro MXML. Para além desses ficheiros é possível ainda criar pastas e ficheiros CSS, de Action Script e de outro qualquer tipo. É possível definir bastantes opções para o projecto, no separador “Properties” do menu “Project”, destacando-se de entre elas a versão do Flash a usar, argumentos opcionais de compilação, referências a outros projectos, versão do Flex SDK a usar, entre outros. No projecto criado foi deixado tudo com o valor por omissão, excepto:

- A versão do Flash Player a utilizar para conseguir correr o programa elaborado, sendo definida a 10.0.0 como mínimo;
- Os argumentos opcionais de compilação, que por omissão vêm com o valor “-locale=en_US,pt_PT” mas que houve necessidade de alterar durante o desenvolvimento, ficando da seguinte forma:

  ```
  -locale=en_US,pt_PT -services "services-config.xml" -source-path=../locale/{locale}
  ```

  A explicação para a adição desses argumentos opcionais está descrita nos próximos parágrafos.

O próximo passo foi elaborar um ficheiro com o nome “services-config.xml” que contém uma estrutura XML capaz de ser interpretada pelo Flex e de criar um canal de ligação a um servidor remoto. Nesse ficheiro foi definido um identificador, para mais tarde ser usado no código em Action Script, e o endereço do servidor de experiências que possui o AMFPHP até ao ficheiro “gateway.php”. Neste caso o endereço definido foi o **http://193.136.60.180:8080/remotelab/amfphp/gateway.php**. Por último foi necessário adicionar aos argumentos de compilação o texto **- services “services-config.xml”** de modo ao ficheiro ser usado na compilação do projecto. Com estas definições no ficheiro “services-config.xml”, a interface em Flex sabe a que IP tem de fazer os pedidos, e em que local do servidor Web se encontra o ficheiro PHP com os serviços.

Um dos requisitos do novo laboratório é o facto deste poder estar disponível em duas línguas, português e inglês. Para cumprir esse requisito foi necessário criar uma pasta chamada “locale” que possui pastas com a sigla de cada linguagem, duas neste caso que foram designadas de “en_US” e “pt_PT”. Dentro de cada pasta foi criado um ficheiro de propriedades, o “Main.properties”, que possui em cada linha uma propriedade correspondente à tradução na respectiva língua. Por exemplo, no ficheiro da língua inglesa foi adicionada a propriedade “MW_RESULTS=Results”, enquanto no de língua portuguesa adicionou-se “MW_RESULTS=Resultados”. Desta forma, quando é necessário mostrar na interface alguma palavra é apenas indicada a referência ao MW_RESULTS, facilitando assim a gestão de línguas em toda a aplicação. Também se poderia ter optado por implementar este sistema de línguas recorrendo a uma base de dados, definindo as propriedades para cada língua numa tabela, mas analisando o tamanho do projecto e o número de propriedades decidiu-se usar ficheiros. Também foi necessário adicionar argumentos de compilação adicionais **-source-path=../locale/{locale}** para que os ficheiros fossem incluídos na compilação do projecto.

Todos os componentes do Flex SDK possuem um aspecto padrão, pertencendo a um tema que vem por omissão com a *framework*, tendo todas as aplicações elaboradas com este tema o mesmo tipo de visual. No entanto, é possível personalizar os componentes visuais de forma a dar a cada aplicação uma identificação visual própria. Dessa forma foi feita uma investigação de temas grátis, disponíveis na Internet, que permitissem omitir o visual por omissão da *framework* com que a aplicação iria ficar. Foi escolhido o tema “Undefined Skin One” do site [Scalenine, 2010], cujo esquema de cores e componentes é muito parecido ao do LabVIEW, tendo no entanto um esquema de acordo com interfaces ricas presentes em várias aplicações da Internet. O tema em si pode ser usado para aplicações não comerciais e é composto por um conjunto de ficheiros CSS, ficheiros de fontes, imagens, ficheiros XML, ficheiros Flash e ficheiros em Action Script, que definem o visual dos componentes. As pastas que possuem todos estes recursos foram copiadas para a raiz da pasta que contém o código fonte, tendo sido feita referência ao ficheiro CSS no ficheiro principal do projecto, de modo a que a aplicação utilizasse a formatação dos componentes indicada no novo tema.

Antes de começar a implementação dos requisitos foi necessário preparar a arquitectura do projecto, isto é, a forma como os ficheiros e o código fonte estão organizados. Optou-se por seguir um rumo que permite ter o projecto organizado, ou seja, seguindo um conjunto de padrões de desenvolvimento de *software* que permita a sua reutilização futura, facilitando desta forma a manutenção do código. Antes de mais, os ficheiros que contêm código fonte foram documentados, tanto os MXML como os ficheiros

Action Script, seguindo as boas práticas recomendadas para projectos em Flex [Adobe, 2010j]. Também se optou por dividir os ficheiros de acordo com a acção que cada ficheiro realiza, usando um conjunto de padrões de construção de *software*, o *Model View Controller* (MVC). O uso do MVC permitiu colocar em directorias diferentes os ficheiros de definição da interface, os ficheiros que tratam do armazenamento de dados, os ficheiros cujo código trata da chamada a serviços, os que lidam com os dados recebidos e os de eventos. Esta separação, tal como foi dito, facilita a manutenção do código pois não existem, por exemplo, ficheiros com código de definição de interface misturados com código de tratamento de dados e da lógica da aplicação.

A implementação de um MVC num projecto sem o recurso a *frameworks* externas pode-se tornar custosa pois é necessário implementar toda a lógica de comunicação entre cada módulo. Por isso foi levantada a hipótese de utilizar uma *framework* que facilite a implementação do MVC [Moses, 2009]. Na investigação efectuada foram detectadas várias *frameworks* que suportam a implementação de MVC. No entanto, sobressaíram quatro delas, que normalmente predominam nas aplicações em Flex, com as seguintes características:

- **Cairngorm** – É bastante popular dado ser uma das primeiras *frameworks* deste tipo a aparecer. É associada muitas vezes à Adobe, pois existem ferramentas oficiais de geração de código;
- **PureMVC** – É bastante popular e possui também implementação para outras linguagens, como Java e C#;
- **Swiz** – Leve, sucinta e bastante clara. Possui maior ênfase e produtividade associada aos ficheiros MXML de definição de interface. No entanto, não tem uma comunidade de utilizadores tão vasta como os seus concorrentes;
- **Mate** – Faz uso de vários padrões de *software* e é caracterizada por oferecer uma elevada flexibilidade.

Não é objectivo desta dissertação fazer uma análise profunda a todas estas *frameworks*, por isso a escolha foi tomada com base no *feedback* encontrado nas comunidades de Flex na Internet. No artigo do site [Hillerson, 2009], por exemplo, foi feita uma análise exaustiva destas *frameworks*, e foi atribuída uma pontuação de um a cinco a uma série de propriedades, em que o valor mais alto representa um melhor funcionamento. O cruzamento dessas propriedades com cada *framework* pode ser observado na Tabela 8.

Tabela 8 - Cruzamento entre as *frameworks* Flex de MVC e as suas características

	Concisão	Familiaridade	Flexibilidade	Poder	Cativante	**Total**
Cairngorm	2	4	4	3	3	16
PureMVC	1	2	5	3	1	12
Swiz	5	5	2	2	4	18
Mate	4	4	4	5	5	22

A *framework* Mate é a que melhor pontuação obteve, destacando-se sobretudo por ser bastante poderosa a nível de flexibilidade. Para além disso possui algumas particularidades em relação às outras, como, por exemplo, fazer uso de vários padrões de desenvolvimento de *software*, destacando-se o *Inversion of Control* (que permite instanciar classes no *runtime* da aplicação), o *Presentation Model* (permite diminuir o código presente em ficheiros de definição da interface) e o MVC clássico. O Mate também faz uso de um dos mecanismos mais poderosos que existe no Action Script 3, os eventos.

Tendo em conta o que foi descrito anteriormente, foi decidido usar a *framework* Mate no projecto. Assim, fez-se *download* da *framework* [Mate, 2010] e colocou-se dentro da pasta "libs", onde normalmente ficam as bibliotecas. De seguida partiu-se para a implementação dos requisitos do projecto, ou seja, o desenvolvimento em si da interface.

O próximo passo foi criar a estrutura de pastas de acordo com o Mate. Numa análise aos requisitos decidiu-se criar a estrutura para o ficheiro de definição da interface principal, que já tinha sido criado no momento da definição do projecto. Optou-se por criar uma estrutura modular, fazendo uso do componente "módulo" existente no Flex. O uso de módulos permite retirar bastante carga à aplicação pois é possível chamar módulos no *runtime* da aplicação, evitando que a aplicação principal fique tão pesada, diminuindo desta forma o seu tempo de carregamento. Com esta estrutura modular, para além de ser necessário a estrutura Mate para o ficheiro de interface geral, foi necessário criar a estrutura para cada módulo, resultando numa melhor organização. Para este projecto apenas foi criado um módulo, que é o da interface de definição dos valores de configuração e de visualização dos resultados, mas se no futuro houver necessidade de elaborar outras funcionalidades é necessário criar outro módulo com a mesma estrutura. De referir ainda que foi criada mais uma pasta na raiz da pasta que contém o código fonte para albergar os componentes partilhados por toda a aplicação, como é o caso de uma janela de espera. A estrutura do projecto explicada nos parágrafos anteriores pode ser consultada mais em pormenor na Tabela 9, que contém a estrutura das pastas, mostrando também a localização dos ficheiros de definição da interface.

Tabela 9 - Estrutura do projecto RemoteLab em Flex a partir da pasta do código fonte

1º Nível	2º Nível	3º Nível
assets		
components		
controller		
css		
event		
front		
manager		
model		
views		
	MainWindow	
		controler
		event
		front
		manager
		model
		MainWindow.mxml
RemoteLab.mxml		

Como se pode observar, o "RemoteLab.mxml" é o ficheiro que é carregado em primeiro lugar quando a aplicação inicia, pois está na raiz da pasta do código fonte. Este ficheiro é o responsável por chamar o único e principal módulo da aplicação, o "MainWindow". Para além disso, também é definido, através da indicação da localização do CSS, o tema a utilizar pelos componentes.

No segundo nível da estrutura de pastas pode-se ver o módulo "MainModule". Caso fosse necessário criar outro módulo tinha-se de criar uma pasta ao mesmo nível que esta, mas com o nome do módulo. Dentro dessa pasta teria de haver uma estrutura semelhante ao que se pode verificar no terceiro nível, exceptuando o nome do ficheiro da interface que deve ser idêntico ao nome do novo módulo.

Tal como foi referido, dentro da pasta "MainModule" foi criado o ficheiro "MainModule.mxml", que contém a definição da interface através dos componentes existentes no Flex SDK, os quais, na sua maioria, são botões, painéis, grelhas, texto, botões de múltipla selecção, estruturas com opções já definidas e os componentes de construção de gráficos. Nos componentes de introdução de valores numéricos foi limitado, para cada campo, a série de valores a introduzir, mostrando-se um balão com os valores possíveis quando o cursor está sobre o campo de texto. Os componentes de visualização de resultados estão escondidos numa primeira instância, sendo mostrados apenas da primeira vez que a aplicação recebe dados. Também é neste ficheiro que foi implementada a lógica de mudança de língua, cujas definições são carregadas no início da execução da aplicação

e sempre que o utilizador muda a língua na interface. Para além disso foram criados vários ficheiros em cada pasta da estrutura, obedecendo a um padrão de nomes sugerida pelo Mate, para implementar toda a lógica de interacção com a interface. Assim, foram efectuadas as seguintes acções:

- Dentro da pasta **front** foi criado o ficheiro "MainWindowFront.as". Neste ficheiro são armazenadas as propriedades com os valores a mostrar na interface. Também contém as funções chamadas quando se dá alguma acção na interface, como por exemplo, quando um botão é clicado. Para além disso, é neste ficheiro que é feito o tratamento das acções realizadas pelo utilizador antes de ser feito qualquer pedido ao servidor, através do disparo de eventos personalizados, que são capturados pelo controlador do módulo;
- Como foi referido anteriormente, são usados eventos personalizados, que no caso do projecto são do tipo da classe "MainWindowEvent.as" criada dentro da pasta **event**. Foram criados três tipos de eventos, um é chamado quando a aplicação necessita de enviar o pedido com os valores de configuração, outro evento serve para questionar o servidor se os resultados já estão disponíveis, o último evento serve para mandar cancelar o pedido. Dentro da classe também existem variáveis para armazenar os valores de configuração e o identificador do pedido, para que o controlador possa aceder a eles e enviar o pedido com os respectivos valores;
- Na pasta **controller** foi criado o ficheiro "MainWindowMap.mxml". Este ficheiro é o controlador da aplicação. Isto é, é ele que se encarrega de capturar os eventos lançados e efectuar as respectivas acções, que neste caso são as de fazer a requisição e a de questionar se os resultados já estão disponíveis. Estas acções são definidas com recurso a componentes específicos do Mate, que oferece mecanismos de captura de eventos e de chamadas a métodos remotos, entre outras funcionalidades. Para além disso, também são definidas propriedades que fazem instanciar alguns objectos e copiar os respectivos valores das variáveis apenas quando recebe os resultados (IoC);
- Na directoria **manager** foi criado o ficheiro "MainWindowManager.as". No controlador é necessário definir, quando um método remoto é chamado, em que locais da aplicação vão ser recebidos e tratados os dados. Assim, é necessário criar tantas funções como chamadas a métodos remotos definidas, que no caso deste projecto foram três: uma para tratar a resposta recebida quando é feita uma nova requisição; outra para tratar os dados quando é

verificado se já existem resultados; e, por fim, outra para tratar a resposta recebida quando é cancelado um pedido. Dessa maneira, é implementada toda a lógica de formatação de dados neste ficheiro. Também foram criadas as mesmas propriedades que as criadas em "MainWindowFront.as", de modo a armazenar os valores obtidos.

Todas as acções que sigam as delineações do Mate devem comportar-se como descrito anteriormente. Ou seja, existe um ficheiro com a definição da interface com o mínimo código Action Script 3 possível, usando o **front** para esse efeito. O **front** lança um **event** que é capturado pelo **controller**, que por sua vez manda efectuar as acções necessárias, como, por exemplo, chamar um método remoto. A resposta da acção é recebida no **manager**, que se encarrega de formatar a resposta. A resposta reflecte-se na interface através da injecção de variáveis do **manager** para o **front**, injecções essas que são definidas no **controller**.

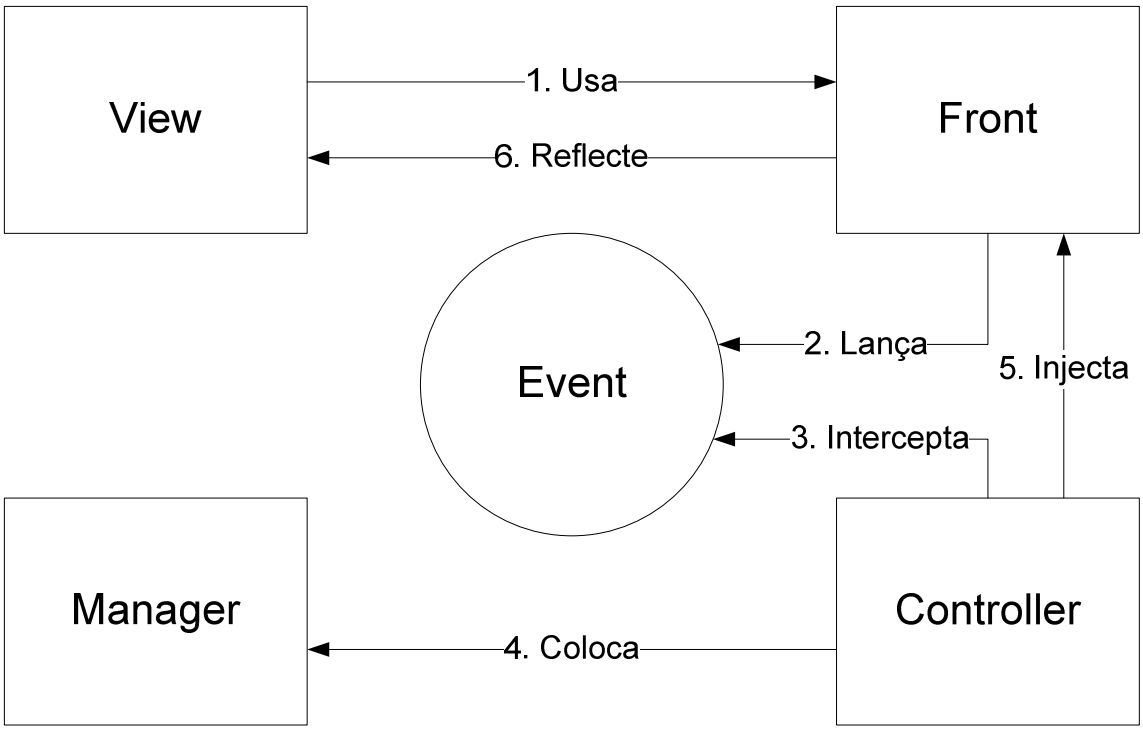

Figura 35 - Ciclo de vida de uma acção em Mate

4.3. Controlo de acessos e segurança

Nesta secção são explicados todos os mecanismos de segurança que foram implementados no novo laboratório remoto. Cada tópico abordado está explicado num subcapítulo em separado.

4.3.1. Controlo de acessos

Uma vez elaborada a interface, foi necessário encontrar uma solução no que diz respeito ao controlo de acessos. Durante o desenvolvimento da aplicação, a interface em Flex foi colocada no servidor de experiências, usando o servidor Apache que foi lá instalado para a disponibilizar para o exterior. No entanto, a aplicação estava disponível num servidor público, tornando-a acessível a partir de qualquer local da Internet. Uma vez que o laboratório remoto se destina apenas a ser usado por um grupo restrito de utilizadores, tipicamente os alunos de uma determinada unidade curricular, decidiu-se implementar um sistema de controlo de acessos que permite evitar o acesso por parte de qualquer utilizador anónimo.

Foram ponderadas algumas opções para implementar o controlo de acessos, como, por exemplo, o desenvolvimento de um módulo de gestão de utilizadores próprio que iria integrar a interface em Flex, ficando deste modo instalado no servidor de experiências. No entanto, esta opção levantaria vários inconvenientes:

- O custo em tempo e esforço de desenvolver um sistema deste tipo é grande. Seria necessário usar uma base de dados e implementar toda a lógica de gestão de utilizadores, isto é: *login*, inserção, alteração e eliminação de utilizadores. Para além disso, seria necessário implementar um mecanismo de permissões pois nem todos os utilizadores podem gerir essa área;
- Os gestores de utilizadores, docentes da unidade curricular, teriam a carga extra de gerir os utilizadores, criando uma conta para cada aluno;
- Os alunos teriam de decorar mais uma palavra passe, para além das inúmeras que já têm de saber no seu dia-a-dia.

Todas estas desvantagens, aliadas ao facto deste projecto não ter como objectivo a construção de um sistema de controlo de acessos, tornaram inviável implementar esta solução. Dessa maneira, ponderou-se a hipótese da interface em Flex ficar alojada no servidor da plataforma Moodle usada pelo ISEP para apoio ao ensino, usando o sistema de controlo de acessos da plataforma, ou seja, implementando um SSO que funciona para o Moodle e para a interface em Flex. O laboratório antigo já usava o Moodle como

intermediário para o controlo remoto do VI, pelo que esta solução ganhou ainda mais força. A interface em Flex é composta pelos ficheiros Flash que são executados pelo *browser*, e funcionam sempre do lado do cliente. Ou seja, para o utilizador tanto faz a localização desde que tenha acesso aos ficheiros. Assim, acedeu-se ao Moodle com permissões de edição e após colocar os ficheiros Flash numa pasta, criou-se um novo recurso do tipo "Apontador para página", em que foi indicada a localização do ficheiro Flash carregado em primeiro lugar. A partir desse momento, todos os utilizadores com permissões de acesso à unidade curricular passam a visualizar um recurso que quando é clicado mostra a interface em Flex. Esta solução permitiu:

- Uma gestão mais eficaz e menos custosa dos utilizadores, pois esta é efectuada pelo administrador do Moodle;
- O uso das mesmas credenciais por parte dos alunos para aceder ao laboratório, Moodle e outros portais académicos;
- Uma solução mais eficaz e rápida, pois evitou integrar a interface em Flex numa plataforma com um sistema de controlo de acessos já estável;
- Registar quem acedeu à interface em Flex através dos registos da plataforma Moodle.

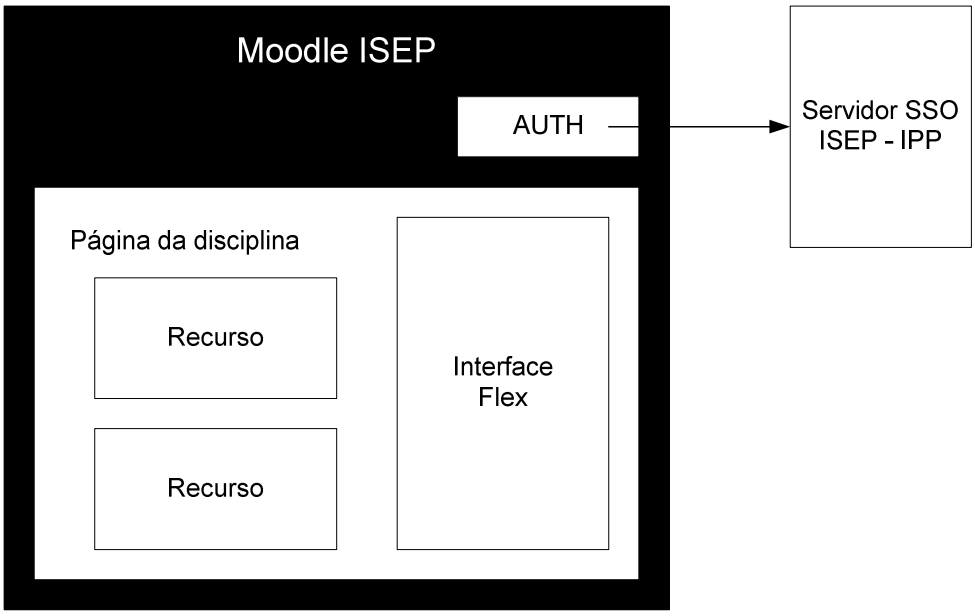

Figura 36 - Arquitectura simplificada do sistema de controlo de acessos

A interface em Flex, quando um utilizador clica no recurso, abre uma nova janela do *browser*, de modo a facilitar a separação da interface da restante plataforma Moodle.

4.3.2. Acesso aos serviços PHP

Os serviços AMFPHP elaborados na linguagem PHP estão disponíveis no servidor Web do servidor de experiências. No entanto, se não for tida qualquer preocupação extra, ficam disponíveis para o exterior. Isto torna a aplicação insegura pois de nada serve o sistema de controlo de acessos, presente no servidor Moodle, se os serviços puderem ser invocados por qualquer cliente. No entanto, existe uma forma de eliminar essa falha, através da definição de um ficheiro "crossdomain.xml" na raiz do servidor Web. Este ficheiro contém uma estrutura XML onde são definidas quais são as aplicações Flash presentes em domínios da Internet autorizados a chamar os serviços em PHP. Como a interface está presente no domínio do Moodle do ISEP, foi adicionado o respectivo endereço ao ficheiro, que ficou com a seguinte estrutura:

```
<?xml version="1.0"?>
<cross-domain-policy>
    <allow-access-from domain="moodle.isep.ipp.pt" />
</cross-domain-policy>
```

Com esta solução evita-se igualmente que alguém faça *download* dos ficheiros Flash e execute a aplicação a partir do seu computador local. Apenas pedidos feitos a partir do domínio do Moodle do ISEP são aceites pelos serviços do AMFPHP. Também foram feitas algumas alterações às definições da plataforma AMFPHP, que permitiram corrigir alguns problemas de segurança de algumas definições que o AMFPHP traz, por omissão, para facilitar o desenvolvimento de serviços. As alterações foram as seguintes:

- Foi eliminado o *Service Browser*. Esta ferramenta permite testar os serviços enquanto estão a ser desenvolvidos, mas se não for eliminado qualquer pessoa pode ter acesso ao nome dos serviços e respectiva localização;
- Foi eliminado o serviço *DiscoveryService* que vem por omissão quando se instala o AMFPHP. Este serviço expunha os métodos disponíveis e os respectivos detalhes para o exterior;
- Definir o AMFPHP para funcionar em modo de produção. No ficheiro "gateway.php" é possível definir se o sistema funciona em modo de desenvolvimento ou em modo de produção através da seguinte linha de código:

  ```
  define("PRODUCTION_SERVER", true);
  ```

Se a propriedade estiver definida como “false”, o AMFPHP lança erros que permitem aos programadores captar excepções ao fazer *debug*. Se o sistema já estiver em produção, esses erros são ocultados, não sendo mostrados a terceiros [TheFlashBlog, 2010].

4.3.3. Código fonte

O novo laboratório remoto é um sistema distribuído onde a implementação da maioria das camadas envolveu a escrita de código fonte, tanto de PHP (na camada intermédia) como de Action Script (na camada de apresentação). Já no que toca à camada de dados, onde foram feitas implementações em LabVIEW, não segue a mesma linha que as linguagens anteriores pois o LabVIEW é uma linguagem de programação gráfica através de componentes já criados. Desse modo, não foi tida em consideração qualquer preocupação com segurança na implementação propriamente dita.

No que diz respeito ao código desenvolvido em PHP, houve uma preocupação em elaborar código consistente, de modo a que o servidor Web seja capaz de executar os ficheiros PHP sem obter nenhum tipo de aviso. Também foi prevista a possibilidade de o utilizador da aplicação injectar código SQL, facto que poderia por em causa a estabilidade do laboratório, para além de acesso a dados não autorizados. Todas as chamadas à base de dados são revistas por uma função que elimina possível código SQL indevido, inserido pelo utilizador.

Em relação ao código desenvolvido em Action Script 3 para construir a interface, a principal preocupação passou por evitar um excessivo consumo de memória através do seguimento das boas práticas da programação, pois a interface vai funcionar do lado do cliente, ou seja, no computador do utilizador. Para além disso, também se verificam os valores a introduzir nos campos de texto, isto é, quando o utilizador introduz um valor de um valor de configuração que não esteja entre os valores possíveis, é avisado de tal facto. Os campos de texto apenas aceitam valores numéricos, efectuando desta forma uma primeira verificação antes de enviar os dados para o servidor.

4.3.4. Outras configurações

Por fim, foram efectuadas algumas configurações que permitiram eliminar as hipóteses de acessos indevidos. No que diz respeito ao servidor Web do LabVIEW, que possui o *Web Service* para ser acedido pelo PHP, foi definido nas opções do servidor apenas aceitar pedidos da mesma máquina. Por omissão, o servidor Web aceita pedidos de todos os IPs, no entanto, não faz sentido o servidor estar aberto para o exterior se apenas vai receber pedidos do PHP para invocar o *Web Service*.

No que diz respeito ao servidor Web Apache, não foi possível uma limitação tão drástica como a do servidor Web do LabVIEW, porque os serviços do AMFPHP têm de estar disponíveis para o exterior, para que os clientes se possam ligar. Já a palavra passe do MySQL, tal como foi explicado num capítulo anterior, foi alterada.

Durante este capítulo 4 foram explicadas em pormenor todas as opções tomadas durante o desenvolvimento, desde as configurações necessárias à implementação, ao próprio desenvolvimento em si nas várias camadas, passando pela preocupação com a segurança da aplicação. A seguir vai ser descrito e mostrado o funcionamento real do novo laboratório remoto.

5. Demonstração de resultados

A implementação descrita nos anteriores capítulos levou a que fossem efectuados vários testes de funcionamento do novo laboratório. De forma a validar o trabalho efectuado, e visto que o laboratório remoto se destina aos alunos da disciplina de Electrónica, foram elaborados testes reais de circuitos baseados em guiões de trabalhos dados nas aulas. A seguir vai ser descrito um dos testes efectuados, que consiste em resolver um guião de um trabalho proposto aos alunos. Também é analisado todo o processo de interacção do utilizador com o novo laboratório, desde que acede ao laboratório até à visualização de resultados.

A estação NI-ELVIS que está ligada ao servidor de experiências, e que trata de obter os resultados do circuito, é a que mostra a Figura 37. Como se pode observar, possui uma placa com um circuito electrónico, o qual vai ser analisado.

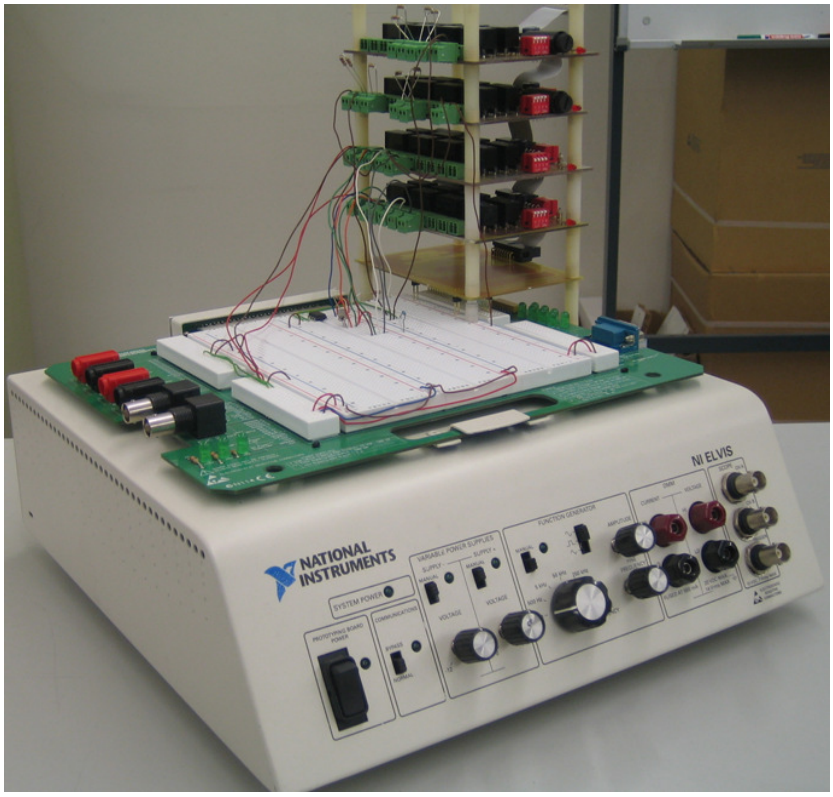

Figura 37 - Estação NI-ELVIS do laboratório remoto

O guião proposto aos alunos pode ser consultado no ANEXO I. Nele, é proposta a análise de um circuito baseado num amplificador operacional, o qual está representado na Figura 38. Como se pode observar, estão demonstradas as diferentes possibilidades de configuração do circuito montado no laboratório remoto. Junto de cada grupo comutador está indicado o endereço e a selecção que o utilizador deve efectuar na interface do laboratório remoto para um ou outro dos componentes ou ramos. As medidas de tensão e corrente são efectuadas automaticamente nos nós e ramos indicados na mesma figura.

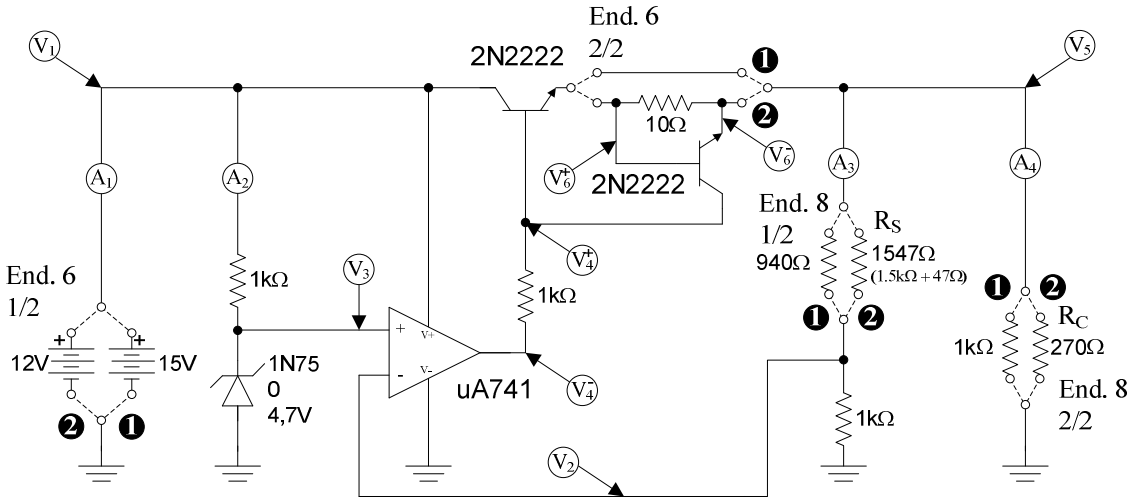

Figura 38 - Esquema do circuito montado no laboratório remoto

O principal objectivo deste guião, é fazer com que os alunos sejam capazes de medir as tensões e correntes, introduzir alterações no circuito e voltar a medir os valores das tensões e correntes, analisando desta forma o comportamento do circuito em diferentes condições de utilização.

Uma vez analisado o guião, é tempo de aceder ao novo laboratório remoto. Assim, é necessária a autenticação no Moodle ISEP, através das respectivas credenciais. Uma vez feito o *login*, acede-se à área da unidade curricular "Electrónica" e clica-se no recurso que abre uma nova janela com a interface do novo laboratório remoto, a qual se pode observar na Figura 39.

De referir que, em relação ao antigo laboratório, houve uma grande melhoria no que diz respeito ao tempo esperado pelo utilizador para o carregamento da interface. O tempo médio de espera situa-se nos 2 a 3 segundos, dependendo das condições de rede, ao passo que no antigo laboratório era necessário esperar cerca de 30 segundos. Para além disso, quando a interface é carregada, permanece na cache do *Web Browser*, o que diminui ainda mais o tempo do seu carregamento nos futuros acessos.

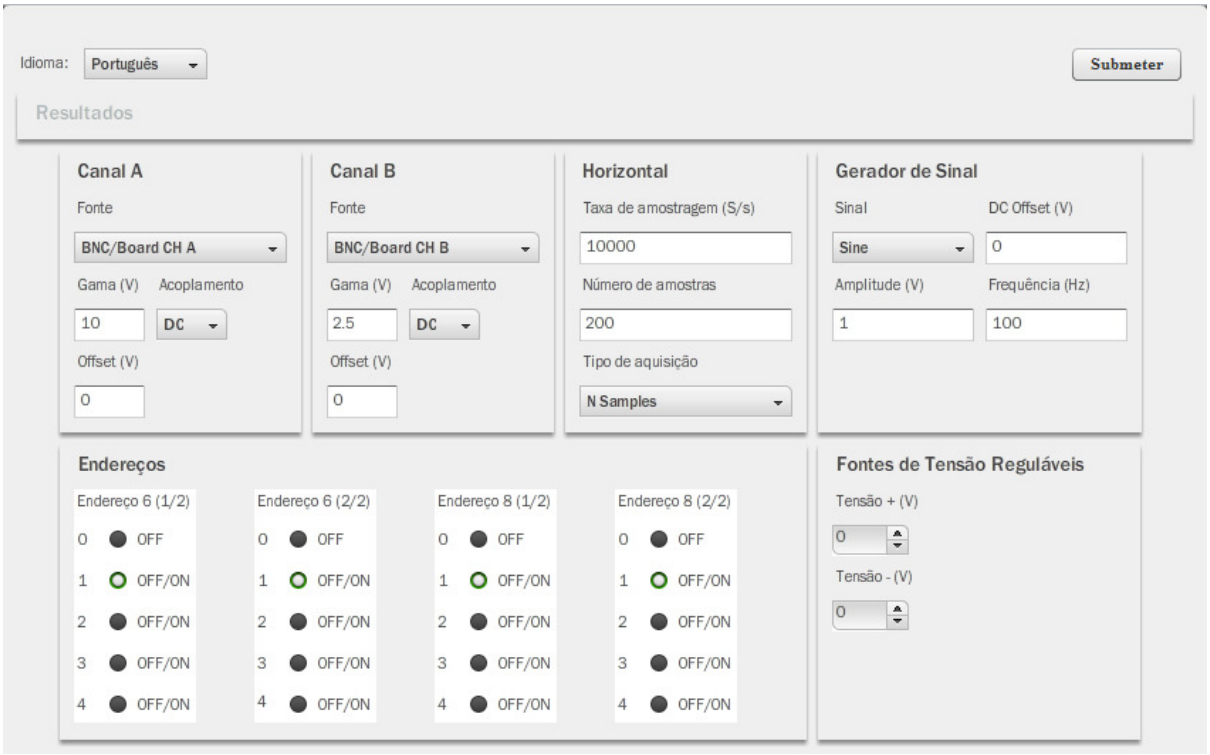

Figura 39 - Interface geral do novo Laboratório Remoto

A interface possui uma zona superior onde se pode alterar a língua da aplicação, que está disponível em português e inglês. No canto superior direito está colocado o botão que faz a submissão dos dados para o servidor. Debaixo da barra de alteração de idioma está presente o painel onde são mostrados os resultados das medições das tensões e correntes, mas devido ao facto de ainda não existirem dados para serem mostrados, o painel está desabilitado e fechado, de modo a não ocupar espaço e fazer com que o utilizador se foque na definição dos valores de configuração. O espaço restante da interface está ocupado por componentes que servem para definir os valores de configuração, que na sua maioria são caixas de texto, botões de múltipla selecção e estruturas com opções já definidas por omissão. Os componentes estão contidos em seis áreas: “Canal A”, “Canal B”, “Horizontal”, referentes à configuração do osciloscópio da plataforma NI-ELVIS, “Gerador de Sinal”, referente à configuração do gerador de sinal da plataforma, “Fontes de Tensão Reguláveis”, referente à configuração das duas fontes de tensão reguláveis da plataforma, e “Endereços”, referente às várias hipóteses de configuração dos módulos de comutação do circuito. O utilizador pode visualizar, num balão, a gama de valores que aceita cada campo de texto quando o cursor do rato está sobre o controlo, como exemplifica a Figura 40 para o caso do valor da “Amplitude” do “Gerador de Sinal”.

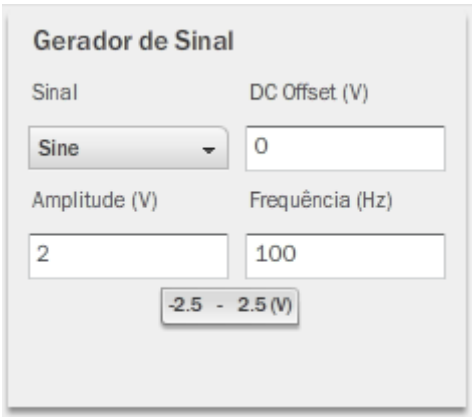

Figura 40 - Balão de ajuda a avisar da gama de valores

Se mesmo assim, o utilizador inserir um valor que não esteja de acordo com o definido para esse campo, é mostrado um alerta a informar que o campo só aceita valores dentro do limite definido, tal como ilustra a Figura 41.

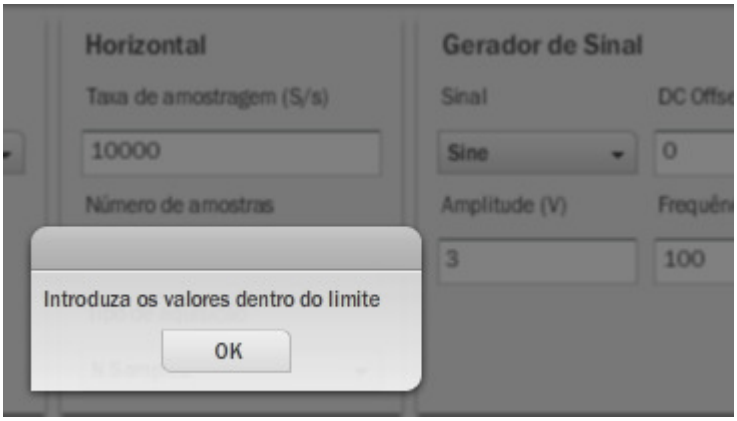

Figura 41 - Aviso a informar de valores inválidos

Quando os valores de configuração estiverem todos definidos, o utilizador pode submeter o pedido, bastando para isso clicar no botão de submissão que se encontra no canto superior direito da interface. Assim, definiram-se todos os valores de configuração, com especial ênfase para os de configuração de módulos de comutação, onde foi escolhido o valor de 15V no “Endereço 6 (1/2)” para a fonte de tensão. Já os valores das resistências a usar, foram definidos no “Endereço 6 (2/2)”, “Endereço 8 (1/2)” e “Endereço 8 (2/2)” os valores 10 Ω, 940 Ω e 1 kΩ respectivamente. Ao submeter o pedido, é mostrada uma janela ao utilizador com a informação para aguardar enquanto o pedido é tratado e não são recebidos os dados, como se pode observar na Figura 42. A restante interface é bloqueada, pois não faz sentido o utilizador definir novos valores enquanto um pedido seu está a ser tratado.

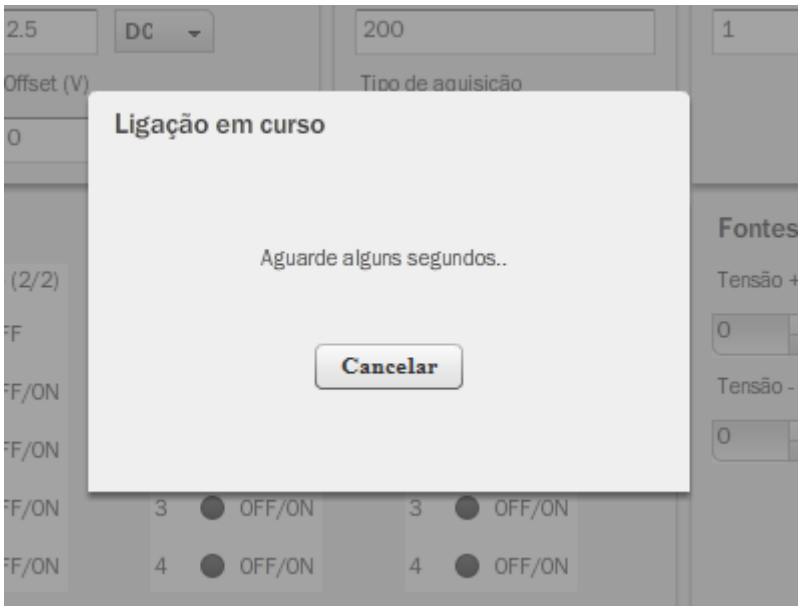

Figura 42 - Aviso de ligação em curso

Quando os resultados estiverem disponíveis, o aviso desaparece e são mostrados os resultados das tensões e correntes medidas nas tabelas e nos gráficos, tal como mostra a Figura 43.

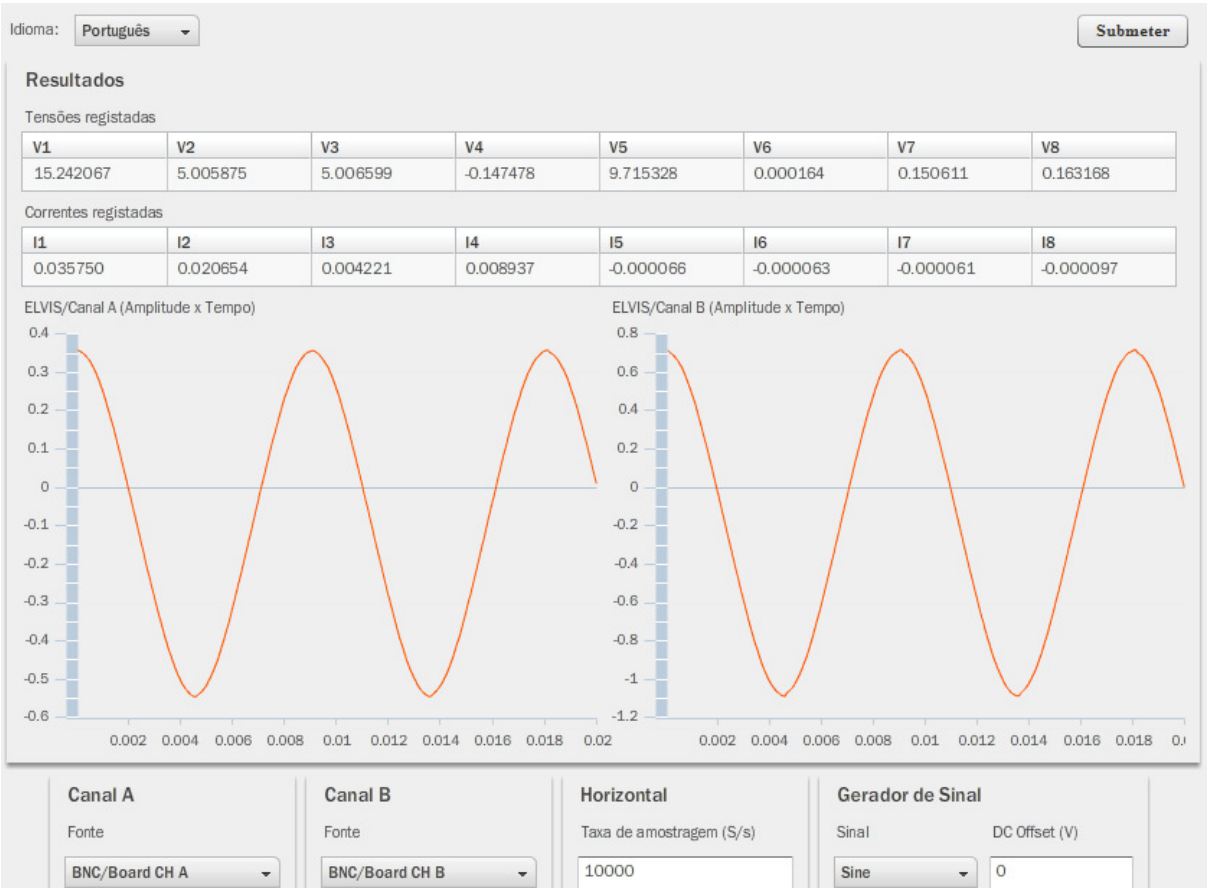

Figura 43 - Interface com os dados apresentados

Após os resultados serem mapeados para as estruturas de visualização, tabelas e gráficos, o painel onde são mostrados é habilitado e aberto, visualizando-se um pequeno efeito ao abrir. Se o utilizador tiver necessidade de voltar a fechar o painel para se focar na definição dos valores de configuração, pode dar um duplo clique sobre o painel que este realiza a acção inversa ao estado em que está, isto é, se estiver fechado vai abrir, se estiver aberto vai fechar. O utilizador pode definir valores de configuração e efectuar submissões as vezes que necessitar, desde que não tenha dois pedidos em simultâneo a serem tratados, tal como foi descrito anteriormente.

Para além da experiência explicada anteriormente, foram efectuados outros testes durante a fase de desenvolvimento, o que permitiu melhorar a interface e corrigir algumas falhas detectadas. A aplicação foi submetida a um teste de carga através do pedido em simultâneo de vários clientes de IPs diferentes e com vários navegadores abertos, tendo o sistema respondido de modo muito satisfatório. Após esse conjunto de testes, pode-se considerar o novo laboratório remoto estável.

6. Conclusão

Neste capítulo resume-se o trabalho efectuado ao longo do projecto e que foi descrito anteriormente, com especial ênfase no que diz respeito à relação entre os objectivos propostos no capítulo 3 e os resultados alcançados.

6.1. Objectivos alcançados

Este trabalho teve como objectivo implementar um novo laboratório remoto que facilitasse o seu acesso aos alunos, isto é, sem necessidade de instalação de um *plug-in* proprietário que só funciona correctamente numa plataforma, reduzindo o tempo de acesso ao laboratório e abstraindo o utilizador do seu funcionamento interno.

No que diz respeito à interface da aplicação, passa a ser possível ao aluno aceder ao laboratório remoto a partir de qualquer sistema operativo e de qualquer *Web Browser*, desde que possua instalado o *plug-in* do Flash Player. Ao contrário do *plug-in* do LabVIEW que variava consoante o sistema operativo e que não funcionava correctamente com todos os *Web browsers*, o *plug-in* do Flash é muito mais transparente e universal. Para além do mais, conseguiu-se ainda um ganho considerável no tempo de carregamento da interface, que passou dos cerca de 30 segundos no laboratório antigo para 2 a 3 segundos no novo, dependendo das condições da rede.

A construção da interface permitiu que o aspecto geral ficasse semelhante à do anterior laboratório, embora com bastantes melhorias a nível de usabilidade. Um exemplo disso é o espaço reservado para a mostragem de resultados, que se encontra escondido e apenas é visível quando os resultados de um pedido estão disponíveis, para que o utilizador se foque na definição dos valores. Para além disso os valores de configuração são validados antes do pedido ser efectuado, evitando a inserção de valores que estejam fora da gama da unidade de medida.

Foi implementada uma lista de espera para tratar os pedidos feitos por vários utilizadores num mesmo intervalo de tempo. Dessa forma, quando o aluno efectua um pedido, este é inserido na lista de espera, ficando a interface encarregue de verificar quando os resultados estão disponíveis. Durante este processo o utilizador pode cancelar o pedido, retirando-o desta forma da lista de espera.

O funcionamento do laboratório tornou-se mais transparente para o utilizador, pois ao contrário do que acontecia anteriormente, apenas existe interacção com a interface da aplicação, a qual é acedida a partir da plataforma Moodle. Por sua vez, a interface ocupa-se de efectuar a comunicação com os restantes componentes do laboratório, abstraindo desta forma o utilizador desse processo.

Para chegar a estes resultados, a arquitectura do novo laboratório foi dividida em várias camadas. A interface, desenvolvida com recurso ao Adobe Flex, comunica através de *remoting* com a camada lógica que está presente no servidor de experiências e foi elaborada em PHP. Por sua vez, esta comunica com uma base de dados em MySQL para a lógica de gestão de pedidos. Também é a partir da camada lógica que são efectuados pedidos ao *Web Service* desenvolvido em LabVIEW 2009, os quais permitem a definição dos valores de configuração e a obtenção dos resultados da experiência. Por fim, o LabVIEW encarrega-se de efectuar a comunicação com a estação NI-ELVIS, que está ligada ao servidor de experiências.

6.2. Trabalho futuro

Ainda no contexto desta dissertação e no sentido de prosseguir com a evolução do trabalho desenvolvido, seria de todo o interesse aproximar a interface da aplicação de modo a assemelhar-se mais a um ambiente real, característica que diminui a resistência dos utilizadores em utilizar sistemas deste tipo. A arquitectura da aplicação está preparada para que a alteração da interface não seja custosa. No entanto, no caso de se optar por este caminho, sugere-se pôr a hipótese de migrar a aplicação para o Flex 4 SDK, actualizando desta forma a tecnologia que suporta a construção da interface.

Para além disso, e ainda no âmbito de sugestões para um trabalho futuro, seria benéfico implementar na aplicação um sistema de histórico de pedidos dos utilizadores. Actualmente, os pedidos efectuados já são guardados na base de dados, necessitando apenas de desenvolver a interface e a lógica de negócio para implementar esta funcionalidade. Assim, seria possível ao docente controlar que alunos acederam ao laboratório, quantos pedidos efectuaram e em que data, quais os valores definidos e quais foram os resultados obtidos. Já os alunos, para além do histórico de pedidos, poderiam comparar e discutir os resultados com outros colegas.

Por fim, seria benéfico para o utilizador poder ver a evolução da lista de espera em tempo real, permitindo ter uma noção mais precisa de quando é que o pedido está a ser tratado e do tempo a esperar.

6.3. Considerações finais

Em suma, ponderados todos os aspectos referidos nos vários capítulos desta dissertação, é possível distinguir este trabalho pela capacidade de abstracção que um utilizador tem de um laboratório remoto, que no seu todo é um sistema bastante complexo com uma arquitectura distribuída.

Foi também possível mostrar que a diferença entre um laboratório tradicional e um laboratório remoto é apenas física, mudando somente a forma como o utilizador acede a ele e define os resultados.

Referências Bibliográficas

Adobe, 2010. “Adobe – Developer Center: Creating marketing platforms with Adobe Flex”. Adobe Systems. http://www.adobe.com/devnet/flex/articles/marketing_platforms_print.html, [acedido em 10 de Julho de 2010].

Adobe, 2010b. “Open source framework, Web application software development”. Adobe Systems. http://www.adobe.com/products/flex/, [acedido em 10 de Julho de 2010].

Adobe, 2010c. “Flex SDK – Adobe Open Source”. Adobe Systems. http://labs.adobe.com/technologies/flex/, [acedido em 10 de Julho de 2010].

Adobe, 2010d. “Adobe Labs – Adobe Flex Framework”. Adobe Systems. http://opensource.adobe.com/wiki/display/flexsdk/Downloads/, [acedido em 24 de Julho de 2010].

Adobe, 2010e. “Free Adobe Flash Platform Technologies”. Adobe Systems. http://www.adobe.com/devnet/flex/free/index.html, [acedido em 24 de Julho de 2010].

Adobe, 2010f. “open source framework, web application software development | Flex”. Adobe Systems. http://www.adobe.com/products/flex/?promoid=BPDEQ, [acedido em 28 de Julho de 2010].

Adobe, 2010g. “Action Message Format 3 Specification”. Adobe Systems. http://download.macromedia.com/pub/labs/amf/amf3_spec_121207.pdf, [acedido em 29 de Julho de 2010].

Adobe, 2010h. “Flex and php development | Adobe Flex”. Adobe Systems. http://www.adobe.com/devnet/flex/flex_php.html, [acedido em 29 de Julho de 2010].

Adobe, 2010i. “The architecture of Flex and PHP applications”. Adobe Systems. https://www.adobe.com/devnet/flex/articles/flex_php_architecture_02.html, [acedido em 29 de Julho de 2010].

Adobe, 2010j. “Coding Conventions – Flex SDK – Adobe Open Source”. Adobe Systems. http://opensource.adobe.com/wiki/display/flexsdk/Coding+Conventions, [acedido em 30 de Agosto de 2010].

AMFPHP, 2010. “Flash remoting for PHP: A responsive Client-Server Architecture for the Web”. Silex Labs. http://www.amfphp.org, [acedido em 29 de Julho de 2010].

Apache Friends, 2010. “apache friends – very easy apache, mysql, php and perl installation without hassles”. Apache Friends. http://www.apachefriends.org/en/index.html, [acedido em 29 de Agosto de 2010].

Balestrino, Aldo; Caiti, Andrea; Crisostomi, Emanuele, 2009. “From Remote Experiments to Web-Based Learning Objects: An Advanced Telelaboratory for Robotics and Control Systems”. University of Pisa, 2009.

Brown, Charles E., 2007. “The Essential Guide to Flex 2 with Action Script 3.0”. Apress, 2007, 25 p, ISBN 978-1-59059-733-0.

Connections, 2010. “LabVIEW Graphical Programming Course”. Connections. http://cnx.org/content/col10241/latest/, [acedido em 8 de Julho de 2010].

Costa, Ricardo, 2003. Infra-estrutura laboratorial para experimentação remota. Faculdade de Engenharia da Universidade do Porto. 2003. Tese de Mestrado.

Dormido, Héctor; Gillet, Chistophe; Esquembre, Francisco, 2009. “Web-Enabled Remote Scientific Environments”. University of Murcia, 2009.

Gassner, David, 2010. “Flash Builder 4 and Flex 4 Bible”. Wiley Publishing Inc, 2010, 7 p, ISBN 978-0-470-48895-9.

Ghoda, Ashish, 2009. “Pro Silverlight for the Enterprise”. Apress, 2009, 3-20 p, ISBN 978-1-4302-1868-5.

Giametta, Chris, 2009. “Pro Flex on Spring”. Apress, 2009, 1-10 p, ISBN 978-1-4302-1835-7.

Hanson, Bem; Culmer, Peter; Gallagher, Justin; Page, Kate; Read, Elizabeth; Weightman, Andrew; Levesley, Martin, 2009. “ReLOAD: Real laboratories Operated At Distance”. University of Leeds, 2009.

Hillerson, Tony, 2009. “FrameworkQuest 2008 Part 6: The Exciting Conclusion”. InsideRIA. http://insideria.com/2009/01/frameworkquest-2008-part-6-the.html/, [acedido em 20 de Agosto de 2010].

James Ward, 2010. “Ajax and Flex Datas Loading Benchmarks”. James Ward. http://www.jamesward.com/2007/04/30/ajax-and-flex-data-loading-benchmarks/, [acedido em 28 de Julho de 2010].

JavaFX Platform, 2010. “JavaFX Overview | Rich Internet applications for Desktop and Mobile”. Oracle Corporation. http://javafx.com/about/overview/, [acedido em 9 de Julho de 2010].

JavaFX Tools, 2010. “JavaFX | Tools”. Oracle Corporation. http://www.oracle.com/us/products/tools/050854.html, [acedido em 9 de Julho de 2010].

JavaFX, 2010. “JavaFX | Rich Internet Applications Development | RIAs java FX”. Oracle Corporation. http://javafx.com/, [acedido em 9 de Julho de 2010].

LABORIS, 2010. “LABORIS : ISEP”. LABORIS. http://www.laboris.isep.ipp.pt, [acedido em 15 de Julho de 2010].

Lewis, N.; Billaud, M.; Geoffroy, D.; Cazenave, P.; Zimmer, T., 2009. “A Distance Measurement Platform Dedicated to Electrical Engineering”. Bordeaux University, 2009.

Mate, 2010. “Mate Flex Framework”. Mate ASFusion. http://mate.asfusion.com/page/downloads/, [acedido em 31 de Agosto de 2010].

Microsoft Silverlight, 2010. “Microsoft Silverlight”. Microsoft Corporation. http://www.microsoft.com/silverlight/, [acedido em 9 de Julho de 2010].

Microsoft Silverlight, 2010b. “Silverlight for Symbian: The Official Microsoft Silverlight Site”. Microsoft Corporation. http://www.silverlight.net/getstarted/devices/symbian/, [acedido em 9 de Julho de 2010].

Moses, Justin, 2009. “The Mate frameworks from a Cairngorm perspective”. InsideRIA. http://insideria.com/2009/07/the-mate-framework-from-a-cair.html/, [acedido em 20 de Agosto de 2010].

MSDN, 2010. “Silverlight Architecture”. Microsoft Corporation. http://msdn.microsoft.com/en-us/library/bb404713%28v=VS.95%29.aspx, [acedido em 9 de Julho de 2010].

National Instruments, 2010. “NI LabVIEW – The Software That Powers Virtual Instrumentation”. National Instruments. http://www.ni.com/labview/, [acedido em 8 de Julho de 2010].

National Instruments, 2010b. “Choose the Right Option for You”. National instruments. http://www.ni.com/labview/how_to_buy.htm, [acedido em 8 de Julho de 2010].

National Instruments, 2010c. “Ni LabVIEW Product family National Instruments”. National Instruments. http://www.ni.com/labview/family/, [acedido em 8 de Julho de 2010].

National Instruments, 2010d. “What is LabView”. National Instruments. http://www.ni.com/labview/optin/whatis, [acedido em 8 de Julho de 2010].

National Instruments, 2010e. “NI ELVIS: Educational Design and Prototyping Platform”. National Instruments. http://www.ni.com/nielvis/, [acedido em 8 de Julho de 2010].

National Instruments, 2010f. “Georgia Institute of Technology Simplifies Teaching Circuit Design with NI ELVIS, NI LabVIEW, and NI Multisim”. National Instruments. http://sine.ni.com/cs/app/doc/p/id/cs-11164, [acedido em 8 de Julho de 2010].

National Instruments, 2010g. “Advantages of Using LabVIEW in Academic Research”. National Instruments. http://zone.ni.com/devzone/cda/tut/p/id/8534, [acedido em 8 de Julho de 2010].

National Instruments, 2010h. “LabVIEW 2009 Features”. National Instruments. http://www.ni.com/labview/whatsnew/features.htm, [acedido em 25 de Julho de 2010].

National Instruments, 2010i. “Converting Vis – NI LabVIEW 8.6 Help”. National Instruments. http://zone.ni.com/reference/en-XX/help/371361E-01/lvupgrade/upgrading_from _previous_ version, [acedido em 25 de Julho de 2010].

National Instruments, 2010j. “Best Practices for Networking with NI LabVIEW”. National Instruments. ftp://ftp.ni.com/pub/events/labview_dev_ed/2009/best.pdf, [acedido em 27 de Julho de 2010].

National Instruments, 2010k. “DAQ Installation/Getting Started Resources”. National Instruments. http://www.ni.com/support/trouble_daq.htm, [acedido em 11 de Agosto de 2010].

National Instruments, 2010l. “Example: Building a Web Service (Windows)”. National Instruments. http://zone.ni.com/reference/en-XX/help/371361E-01/lvhowto/web_ service_ex, [acedido em 12 de Agosto de 2010].

National Instruments, 2010m. “URL Mappings Page (Web Service Properties Dialog Box)”. National Instruments. http://zone.ni.com/reference/en-XX/help/371361E-01/lvdialog/ routing_template_rws_page/, [acedido em 12 de Agosto de 2010].

National Instruments, 2010n. “Interacting with Web Method VIs (Windows)”. National Instruments. http://zone.ni.com/reference/en-XX/help/371361E-01/lvconcepts/ interacting_vis_ws/, [acedido em 12 de Agosto de 2010].

RIASTATS, 2010. “Rich Internet Application Statistics”. RIASTATS. http://www.riastats.com/, [acedido em 9 de Julho de 2010].

ScaleNine, 2010. “ScaleNine : Gallery”. ScaleNine LLC. http://www.scalenine.com/ gallery/index.php, [acedido em 21 de Agosto de 2010].

Silverlight, 2010. “Home: The Official Microsoft Silverlight Site”. Microsoft Corporation. http://www.silverlight.net/, [acedido em 9 de Julho de 2010].

Sousa, Nuno, 2008. Bancada Laboratorial Remota para o Ensino de Electrónica. Instituto Superior de Engenharia do Porto. 2008. Tese de Mestrado.

Sousa, N.; Alves, A.R.; Gericota, M.G., 2010. “Na integrated reusable remote laboratory to complement electronics teaching”, IEEE Transactions on learning technologies, Vol. 3, Nº 3, July - September 2010, pp. 265-271.

StatOWL, 2010. “Rich Internet Application (RIA) Market Share / Global Usage”. StatOWL. http://www.statowl.com/custom_ria_market_penetration.php, [acedido em 9 de Julho de 2010].

TheFlashBlog, 2010. “TheFlashBlog >> AMFPHP Security Basics”. ThaFlashBlog. http://blog.theflashblog.com/?p=419, [acedido em 24 de Agosto de 2010].

TIOBE, 2010. “TIOBE Software: Tiobe Index”. TIOBE Software. http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html, [acedido em 28 de Julho de 2010].

Weaver, James L.; Gao, Weiqi; Chin, Stephen; Iverson, Dean, 2009. “Pro JavaFX Platform”. Apress, 2009, 1-7 p, ISBN 978-1-4302-1876-0.

ANEXOS

ANEXO I – Guião do trabalho proposto aos alunos

Estudo de uma aplicação baseada no uso de um amplificador operacional

O objectivo deste trabalho é que o aluno, perante um circuito baseado num amplificador operacional, seja capaz de analisá-lo e de descrever a sua funcionalidade. A planificação proposta pretende dotar o aluno da metodologia e ferramentas de análise semelhantes às utilizadas em ambiente industrial. O trabalho está dividido em três etapas:

1. Análise do circuito alvo considerando o modelo ideal do amplificador operacional;
2. Verificação do funcionamento do circuito em ambiente de simulação OrCAD;
3. Verificação do funcionamento do circuito através da implementação real.

No final deve apresentar um relatório que permita verificar que foram cumpridas todas as etapas intermédias do trabalho.

Proposta de trabalho

Circuito alvo

O circuito alvo deste trabalho está representado na figura 1.

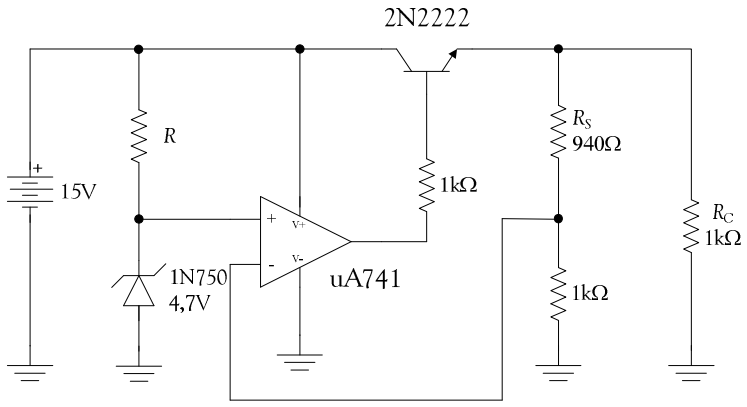

Figura 1: Circuito proposto para análise

1ª etapa - análise teórica do funcionamento do circuito usando o modelo ideal do amplificador operacional

Na primeira etapa do trabalho deve:

1. Determinar o valor da resistência *R* tendo em conta o valor da tensão de alimentação e o correcto funcionamento dos componentes do circuito e considerando que tem apenas disponível a série E12 de resistências (1,0, 1,2, 1,5, 1,8, 2,2, 2,7, 3,3, 3,9, 4,7, 5,6, 6,8, 8,2 e 10,0);
2. Analisar o circuito alvo utilizando o modelo do amplificador operacional ideal, determinando (se possível) todos os valores da tensão em todos os nós do circuito e das correntes em todos os seus ramos operacionais;
3. Repetir os cálculos alterando o valor de R_C de 1 kΩ para 270 Ω;
4. Variando o valor da resistência R_S de 940 Ω para 1547 Ω (= 1,5 kΩ + 47 Ω), repetir a alínea 2 e 3;
5. Variando a tensão de entrada para 12 V, repita as alíneas 2 a 4.

Considere agora o circuito representado na figura 2.

6. Descreva a alteração introduzida no circuito e qual a sua funcionalidade;
7. Para o circuito da figura 2, e exclusivamente com a tensão de alimentação de 15V, recalcule os valores da tensão e da corrente na resistência R_C para o valor de 1 kΩ e para 270 Ω;

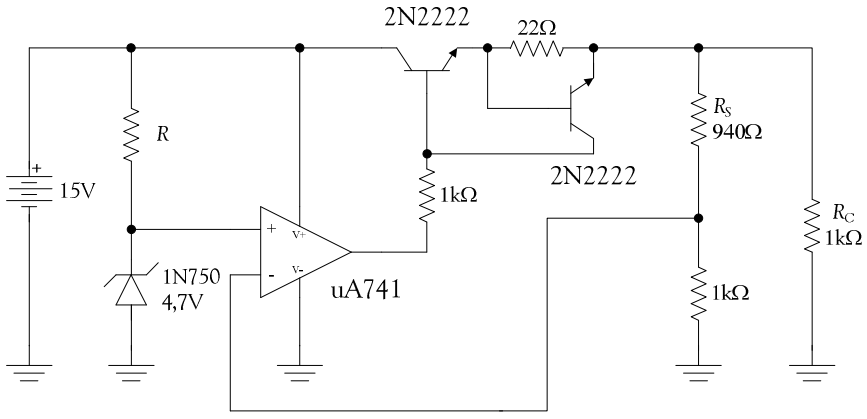

Figura 2: Alteração ao circuito inicial

8. Altere o valor de R_S de 940 Ω para 1547 Ω (= 1,5 kΩ + 47 Ω), e repita a alínea anterior;

9. Calcule o valor máximo da corrente que pode ter a alimentar a carga e o valor mínimo da resistência de carga para que a regulação não se perca para os dois valores de R_S.

Considere agora o circuito representado na figura 3.

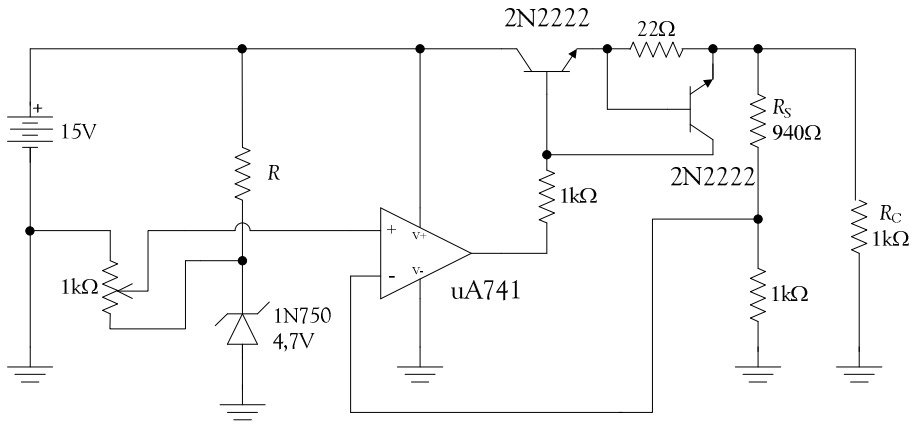

Figura 3: Segunda alteração ao circuito inicial

10. Descreva a alteração introduzida no circuito, qual a sua funcionalidade e as limitações que apresenta, quantificando-as;
11. Por último, descreva a totalidade do funcionamento do circuito e qual o seu objectivo.

2ª etapa – Simulação

A segunda etapa do trabalho corresponde à verificação do funcionamento dos circuitos das figuras 1 e 2 em ambiente de simulação utilizando a aplicação OrCAD.

12. Verifique na simulação a totalidade dos resultados teóricos que obteve na parte analítica, explicando cada resultado não coincidente; se não conseguir a convergência dos valores, seleccione em *PSpice → Edit Simulation Profile* a opção *Skip the initial transient bias point calculation*; explique o porquê da não convergência.

3ª etapa - Montagem

A terceira etapa do trabalho corresponde à verificação do funcionamento do circuito através da sua implementação prática.

13. Monte o circuito da figura 1;
14. Verifique a concordância entre a totalidade dos valores teóricos e simulados que anotou e os que obtem experimentalmente; para a medida das correntes use o método indirecto;
15. Repita a alínea anterior mas agora com uma resistência R_C de 270 Ω;
16. Varie R_S para 1547 Ω (= 1,5 kΩ + 47 Ω) e repita as duas alíneas anteriores;
17. Variando a tensão de entrada para 12 V, e para os dois valores da resistência R_S (940 Ω e 1547 Ω), obtenha os valores de tensão e corrente na resistência R_C para o valor de 1 kΩ e para 270 Ω;
18. Altere o circuito para a configuração da figura 2 e exclusivamente com uma tensão de alimentação de 15 V repita as alíneas 14 a 16.

Utilização do Laboratório remoto

O circuito aqui descrito encontra-se disponível remotamente através da página da disciplina no Moodle. O Laboratório remoto não é um substituto da aula prática de montagem, mas complementar.

O circuito montado no Laboratório remoto é uma réplica daquele que montou no laboratório e, por isso, pode terminar o trabalho iniciado na aula prática, verificar as medidas que já efectuou e estudar as alterações no seu funcionamento, seleccionando as 16 possibilidades diferentes de configuração disponíveis.

Na figura 4 estão demonstradas as diferentes possibilidades de configuração do circuito montado no Laboratório remoto. Junto de cada grupo comutador está indicado o endereço e a selecção que deve efectuar no interface do laboratório remoto para um ou outro dos componentes ou ramos. As medidas de tensão e corrente são efectuadas automaticamente nos nós e ramos indicados na mesma figura.

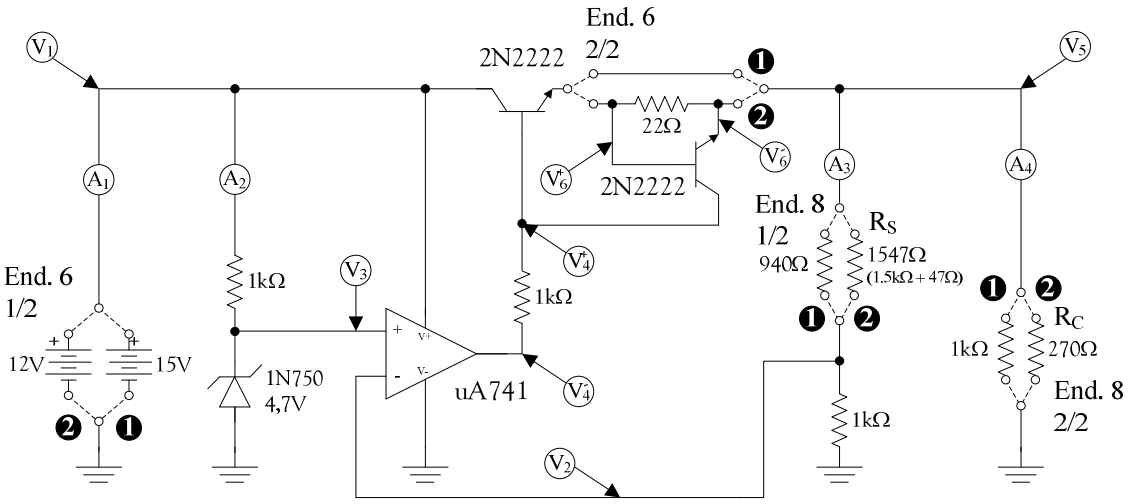

Figura 4: Circuito montado no Laboratório remoto

Relatório

Os alunos poderão esclarecer dúvidas e pedir orientações para a realização dos relatórios durante as aulas de laboratório e de tutoria.

O relatório deve conter:

1. As descrições e os cálculos analíticos pedidos nas alíneas 1 a 11;
2. Os valores que retirou das várias simulações que efectuou; **uma vez que as grandezas são invariáveis no tempo, não há necessidade nem acuidade na colocação no relatório de gráficos temporais;**
3. Uma análise comparativa entre os valores analíticos e os valores obtidos na simulação, justificando eventuais discrepâncias;
4. As leituras efectuadas durante a montagem;
5. Uma análise das discrepâncias que eventualmente possam existir entre o estudo analítico, a simulação e os resultados da montagem;
6. Uma tabela comparativa contendo todos os resultados obtidos nas três etapas do trabalho – cálculos teóricos, simulação e montagem;

Por último, uma análise dos resultados associada à descrição de cada uma das montagens, comentando de que forma as alterações introduzidas nas várias alíneas (alterações nos valores de várias resistências, na tensão de alimentação e introdução de novos componentes) influenciaram ou alteraram o comportamento do circuito e a sua funcionalidade.

ANEXO II – Anexos digitais

Como complemento a esta dissertação, segue em anexo um suporte digital para os seguintes elementos:

- Exemplar da dissertação em formato PDF;
- Projecto Adobe Flex 3;
- Projecto PHP;
- Projecto LabVIEW 2009.